Find & Replace It!

Version 1.4.5

User Manual

Last update: May 2011

Table of Content

1 About Find & Replace It!

1.1 Summary

Find & Replace It! is a high-end solution to find files and execute search and replace operations across multiple files. It allows performing very complex batch replacements inside text files of any size. It supports regular expression syntax and dozens of encodings. It has scripting capabilities which allow transforming on the fly the replacement text for every found string. It even handles batch processing of the encoding of files, as well as of types of end-of-lines.

Why and when do you might need to use it? Well, have you ever had this one piece of text you wanted to modify in hundreds of files at the same time (like that copyright in your php/jsp/cpp/java file or that URL in your html files)? No more need to open text files one by one to edit them! *Find & Replace It!* can replace any contents inside of many text files in just a click. It let you filter files to process according to your wishes, then you only need to enter the text to find and the text to replace to start the replacement. It provides statistics about the number of replacements within each individual file, and counts the number of processed files. Of course, all classical features of such a tool like the possibility to save your job, backup modified files, export the replacement report, and much more are available.

So far *Find & Replace It!* looks like a classical search & replace utility like many others on the web. However here are five key points that make the difference with its competitors:

- Handles more than 50 different kinds of text encodings. If you are stucked with any UTF-8 encoding, any Chinese contents, or any others non ASCII encodings, *Find & Replace It!* is the solution for you. It also handles the different kinds of end-of-line.

- Handles huge files. If you want to process any kind of files containing tons of text, like a log of 10 GB, *Find & Replace It!* will do it for you.

- Provides a regular expression editor. This editor offers tools to easily build advanced regular expressions even if you are a newbie.

- Provides a preview for found and replaced text. The preview window gives you an immediate feed-back that makes easy to check the impact of your replacement within any file without modifying its content. This is especially useful whenever you are searching for a complex expression and you don't want to replace it blindly in many files. This is of course an excellent regex tool, that let's you easily test your regular expressions.

- Makes it possible to dynamically adjust the replacement text. It provides many ways of dynamically adjusting the replacement text according to the context. For instance, it is possible to reuse a fragment of the found expression into the replacement text, do arithmetic operations on found text, insert the path or the name of the processed file, apply conditional operations on the replacement string (JavaScript interface), manipulate dates, etc.

These key features associated to many others makes it one of the most feature rich tool to find and replace regular expressions over multiple files. It's also a powerful tool for converting the text encodings (charsets conversion, including the Byte Order Marks), or the end-of-line delimiters, across multiple files. In addition, it's probably the only software in this category that is portable!

These key features associated to many others makes it one of the most feature rich tool to find and replace regular expressions over multiple files. It is also a powerful tool for converting the text encodings (charsets conversion, including the Byte Order Marks), and the end-of-line delimiters, across multiple files. In addition, it is probably the only software in this category that is portable!

The GUI of *Find & Replace It!* consists of seven panes that you can arrange in any way you want. You can dock panes side by side, arrange them in tabs, or make them float. Adjust the windows

layout to the way you like to work. Find & Replace It! comes with four pre-arrange layouts that can be customized the way you want depending on the size of your screen. You can switch from one layout to another in a simple click.

1.2   Main Features

The list below describes some of the most important characteristics of *Find & Replace It!*:

- Find and replace across many files at once
- Supports regular expression syntax
- Allows multi-line matching
- Supports many text encodings, including Unicode (e.g. UTF-8)
- Preserves line endings while processing files
- Preserves BOM while processing Unicode files
- Allows you to perform dynamic replacements based on found expression captures
- Provides built-in processing function for dynamic replacements (e.g. convert captured expressions to lower case, Base64 encoding, Hex encoding, UTF-8 encoding, etc.)
- Provides a JavaScript like interface to customize replacements on the fly by script processing
- Displays matched expressions reports for file search/replace operations
- Full featured dynamic preview of matched expressions and replacements
- Provides tools for converting text encoding
- Provides tools for converting line endings (Windows, Unix, Macintosh, Unicode)
- Detects text encoding and line endings of files
- Provides advanced filtering options for selecting files that need to be processed, including file name filters and file path exclusion filters
- Allows you to load and save expressions to find, replacement definitions and file filters
- Handles huge files (> 10 GB)
- Regular expression editor / tester
- Fully multi-threaded for fast processing and responsiveness
- Allows you to cancel long operations
- GUI is totally modular
- Creates backup of changed files if required
- Exports the search and replace reports
- Cross-platform: Windows, Mac OS X and Linux

1.3   Supported Platforms

Please refer to the appropriate installation instructions available at:

http://www.dprog.ch/multimedia/findreplaceit-docs/1.4/install.html#title-supported-platforms

The supported platforms varies over time. Therefore you are kindly requested to use the appropriate link, with the appropriate version number (findreplaceit-docs/<version>/install.html), to get the correct notice.

1.4   Getting Started

The installation instructions are available here.

Find & Replace It! is protected by a licenses system. Hence to get a license for the full version of the product you should activate your product with a serial code called *Activation Key*. Without the activation, the software can still be run in demo mode with some limitations. You'll find more information in the Activation chapter.

In order to quickly get started with the main components of the GUI, here is a 2-minutes guide.

All references to the online documentation and resources are listed on this documentation page.

1.5   Support and Services

For general information, please visit our website at: http://www.dprog.ch

The chapter End User License describes the licensing terms for *Find & Replace It!*. If you have any questions about pricing and/or license terms, don't hesitate to write to us at: order@dprog.ch

All support requests regarding software usage as well as general questions about demo version must be addressed to: support@dprog.ch

Please note that support might be only available to registered customers or users who have a valid license for their software copies. Moreover we kindly request our customers to use the online form for posting support requests. This form is accessible through:

http://www.dprog.ch/home/support/

Any User who has a valid license is eligible for free support services.

Finally, you might read the online Terms of Use statement for details about services provided by dProg – Philippe Docourt.

1.6   What to Do in Case Of Problems?

In case you encounter a reproducible crash, we might request you to trace the application's activity. This can be done by running *Find & Replace It!* from the command line with the `-trace` argument. This produces a trace file that might help us to identify when the crash occurs.

In order to trace the application activity you should go through the following steps:

1. Open a terminal
2. Go to the software installation directory: `cd <path/to/FindReplaceIt/binary>` (on Mac OS X: `cd '/Applications/Find & Replace It!.app/Contents/MacOS'`
3. Type: `./FindReplaceIt -trace [<path/to/trace/file>.xml]`

The last command enables an advanced logging for debug purpose in case of problems. Without the optional argument that specify a custom path for the trace file, this command will generate by default a trace file named `trace.xml` alongside the application executable file (on Mac OS X: `'/Applications/Find & Replace It!.app/Contents/MacOS/trace.xml'`). This file contains absolutely no personal information about you, but data about internal running of the software. We kindly request you to send us this trace file called `trace.xml` at support@dprog.ch, preferably in a compressed format.

Important Note: If you do not have write permission into your installation directory, use the additional parameter, which comes after the `-trace` argument, in order to redirect the output into a directory that is writable. Alternatively you might copy/paste your installation into a folder where you can have write access to be sure the trace file will be generated.

1.7   Known Issues and Limitations

The text-encoding detection uses heuristics that do not always provide accurate results except for all Unicode encodings (i.e.: UTF-8, UTF-16, UTF-32). Since version 1.0 the charset detection has been greatly improved. Since this version we use the ICU library from IBM. However, the process is partly statistical in nature, and the results can not be guaranteed to always be correct.

For performance reasons the ***Find & Replace Preview*** has a content limit of 100'000'000 characters. By default this limit is fixed to 5'000'000 to increase search speed. This can be changed through the preferences panel.

For performance reasons it is not possible to search for an expression longer than 50'000'000 characters. By default this limit is fixed to 5'000'000 to increase search speed. This can be changed through the preferences panel.

The product activation requires an Internet connection. There is no way to proceed with activation by phone or any other communication channel.

For the sake of performance, the end-of-line detection reads a maximum of 10 MB of the analyzed file.

The file creation date is not preserved when a file is modified by a search and replace operation.

It is not possible to do batch replacements directly from the command line yet.

The product activation requires an Internet connection. There is no other way to activate your copy of the software.

Unfortunately, *Find & Replace It!* cannot handle MS Word documents, Pages, PDF or others. It is very powerful to manipulate text files, but is limited to thereto. The binary documents like those of MS Word are usually described in a proprietary format. For example, it is not directly possible to extract the text content of a paragraph from a Word document. Indeed, in this case, the text of the paragraph does not appear in clear text in the contents of the DOC file. To get the plain text one must first process the document and the requested manipulations are not documented by Microsoft. However, even if it did, these manipulations are extremely complex (formatting, tables, etc.) and such a feature will far exceed the range of a simple utility software.

2   Installation

Please refer to the appropriate installation instructions available at:

http://www.dprog.ch/multimedia/findreplaceit-docs/1.4/install.html

These instructions might vary from a version to another. Therefore you are kindly requested to use the appropriate link, with the appropriate version number (findreplaceit-docs/<version>/install.html), to get the correct installation notice.

The installation notice covers the supported platforms, how to install the software under Windows, Mac and Linux.
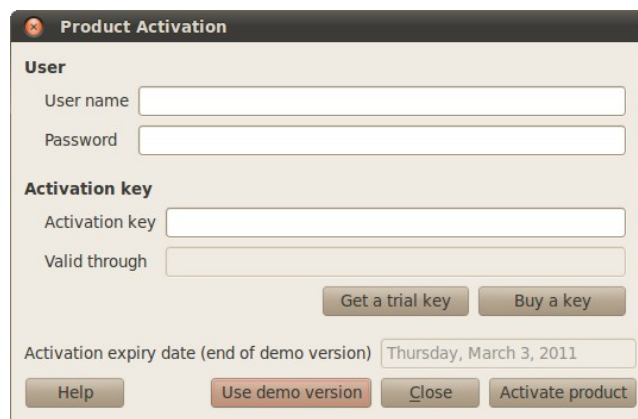
3   Activation

There are two ways of using *Find & Replace It!*, either in demo mode or in full-featured mode. Every time you start the application, a dialog window will ask you to activate your copy of *Find & Replace It!* or to run it in demo mode.

Note: For people who bought *Find & Replace It!* on the App Store from Apple, there is no activation key required to use the software, but there is no demo version available either. In that case this chapter does not concern you.

3.1   Demo Version
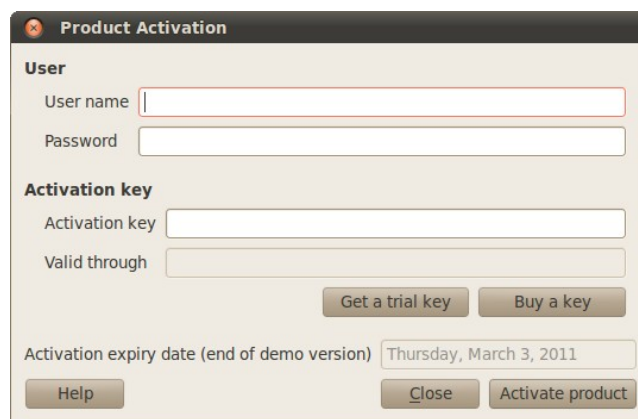
In demo mode, you don't need to proceed with activation, meaning you don't need an activation key, or a login on our website.

The demo version comes with all major features except that you can neither save configuration files nor replace text directly in files. However, the preview window let you see the result of replacements in a read-only mode.

The demo version is available until March 3, 2011.

The activation window shows you an expiry date for activating the software. After this date, the software will not start any more without being activated first. Until this date, you can simply refuse the activation by pressing **Use demo version** and use the software in demo mode. Until this date, you can simply refuse the activation by pressing **Use demo version** and use the software in demo mode. Once that the demo period expired the button is replaced with a button called **Close**.

The demo period ended on March 3, 2011. There is no other choice than activating the product.

3.2 Full Version

The activation process registers your activation key, so that your personal login will own it. An activation is necessary to gain access to the full version of the product.

The activation requires the following steps:

1. Enter your login for www.dprog.ch website (i.e. you have to register first on our website);
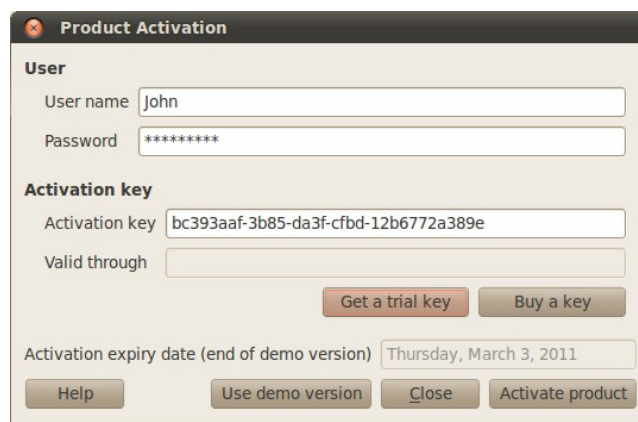
2. 1. Get an activation key or recover a previous activation key:

    1. You can get a 30-day trial key for activating your copy by clicking on **Get a trial key**.

    2. You can buy an activation key by clicking on **Buy a key**. You should receive your activation key by e-mail once you have bought it.

    3. You can recover the last activation key used on your computer (e.g.: after a new install) by clicking on **Get a trial key**. You should then get a message announcing that your activation key has been synchronized. This works either for a trial key or a perpetual key.

The previous activation key used was recovered by clicking the button Get a trial key.

3. Type or paste your activation key in the appropriate field;

4. Press **Activate product** and wait for the answer.

    1. The activation process requires a connection to Internet.

    2. In case of success, the activation window will be automatically closed. Otherwise, you might try again later. In case of problem contact our support.

Once that the activation has been done there is no more possibility to use the demo version.

Note that if the field named **Valid through** is empty, then your license will last forever otherwise your activation will be revoked by the date shown in this field.

All activation keys refers to some hardware identifiers once they have been activated, hence it is not possible to use the same activation key on several computers. In the same way, a trial key is unique for a certain hardware and thus it is not possible to obtain several trial keys for the same computer.

Once the product has been activated, the activation dialog no longer shows up at startup time. However, this dialog might reappear from time to time if your hardware or software configuration has changed. In this case, you will simply have to reactivate your key. This will automatically update your hardware ID in our database. If too many changes occur, the activation might be refused. In such a case we kindly request you to contact us in order to reactivate your copy of the software.

You can check your activation key at any time through the *Help/Activate product...* menu as shown below:
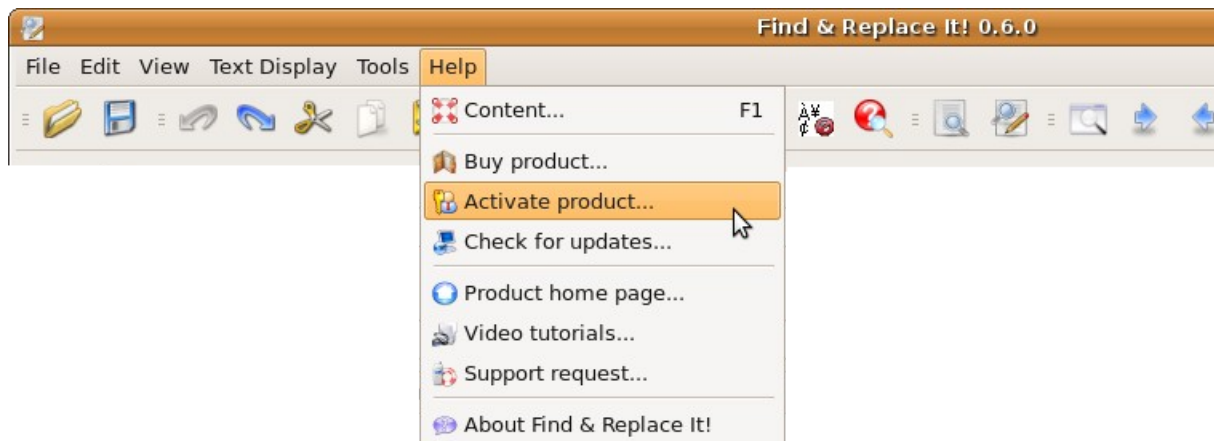
4   User Interface

The GUI of *Find & Replace It!* consists of seven panes that you can arrange in any way you want. You can select which component must be displayed as central widget; you can dock panes side by side, arrange them in tabs, or make them float. Adjust the windows layout to the way you like to work. This enables you to be at ease on a wide range of screen resolutions, from your laptop with its 13.3-inch display up to your 30-inch display.

Find & Replace It! comes with three pre-arrange layouts, one for each possible central widget, that can be customized the way you want depending on the size of your screen. You can switch from one layout to another in a simple click.

Three panes can take place in the center of the main window:

- ***Files Selector***;
- ***Find & Replace Editor***;
- ***Find & Replace Preview***.

When toggled, each of these panes becomes the center of a pre-arranged layout that can be customized. The left side bar of the main window provides a shortcut for toggling the visibility of these windows.

To have the ***Files Selector*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing ***Ctrl+F1***.

To have the ***Find & Replace Editor*** as the central widget. you have to select:
The visibility of this central widget can be toggled by pressing ***Ctrl+F2***.

To have the ***Find & Replace Preview*** as the central widget, you have to select:
The visibility of this central widget can be toggled by pressing ***Ctrl+F3***.

This central widget can't be moved, but it can be resized (to become more or less large and more or less high).

The top, bottom and right (not left) areas around the central widget are classical dock areas for dockable widgets.

To toggle the visibility of the windows that you want to have on the screen, you have to check mark some of the seven icons located on the left side bar of the main window. These seven icons are shown below:

Files Selector	***Find & Replace Preview***	***HTML Viewer***	***Find & Replace Editor***
Shortcut: **Ctrl+1**	Shortcut: **Ctrl+2**	Shortcut: **Ctrl+3**	Shortcut: **Ctrl+4**

Regular Expression Editor	***Find Text***	***Output***
Shortcut: ***Ctrl+5***	Shortcut: ***Ctrl+6***	Shortcut: ***Ctrl+7***

Every pane (also called widget, panel, or window) can be floating, stacked one on the others or just docked.

As you can see, the three central widgets of the three pre-configured layouts are also available as dockable widgets, when one of the other two pre-configured layout is selected. In other words, when a particular window is displayed as a central widget, it is not any more available as a dockable window. This simply means you can't have twice the same window on the screen.

If you have two or more stacked widgets on the others in the same dock area, you can see them as tabs at the bottom of the area.

For example, when the *Find Text* is stacked on the *HTML Viewer* you get that:

5   Two-Minute Guide to Replacing Text in Files

1. Show the File Selector window.

2. Select the folder in which to search for files.

Search in ndreplaceit/src/res/examples/replacemen

☑ Search in subdirectories

File name filter *.txt;*.csv;*.dat;*.ini

3. Enter the appropriate filters to search for files.

	File path exclusion filter	Filter syntax	Case sensitive
1	/images/	Simple text	☐

4. Check mark all files to process (search, replace, convert encoding, etc.)

Found files

Filter _____ Syntax Simple text

☑ Case sensitive filter ☑ Case sensitive sort

	File path	Encoding
1	☑ /Volumes/data/de...ches.preview.csv	ISO–8859–9
2	☑ /Volumes/data/de...tern.preview.txt	wind...1252
3	☑ /Volumes/data/de...ring.preview.txt	UTF–8

☐ File encoding

☑ File backup

☐ Backup modified files   Backup suffix back

5. Tell whether you want to backup changed files or not.

6. Show the Find & Replace Editor.

Expression to find

Find Find & Replace It!

Syntax Simple text   ☐ Case sensitive ☐ Whole word ☐ Minimal match

7. Enter the expression to search for.

☑ Replacement expression

Replace with Find and Replace It!

☑ Capture processing   ☐ Script processing

	Capture available	Capture processing
1	☐	No process for capture

8. Enter the expression to replace with.

5	☐	No process for capture
6	☐	
7	☐	
8	☐	
9	☐	

10. Click on the appropriate button to start searching or replacing text in files.

9. Verify the matched expressions in the preview.

11. Show the Output window.

12. Verify the statistics on texts found or replaced.

Display mode Found and replaced expres Occurence 1/13

Encoding UTF–8

Output Find & Replace Preview

6   Functions

6.1   Selecting Files to Process

The files to be searched and / or converted can be selected through the panel called **Files Selector**.

There are two sections that work together for searching and selecting files. They are **Search scope** and **Found files**.

1. Select a folder to search in. This is the root path where you want to scan for files. You can type a path directly in the **Search in** text field or use the button on the right.
2. Choose to search files recursively into sub folders or not by toggling the **Search in sub-folders** check mark. When searching recursively you can choose to include the hidden folders or not with the **Include hidden and system folders** check box.
3. Enter zero, one or many file name filters within the **File name filter** field. These filters interpret wildcard characters like '*'. They must be comma separated.
4. Optionally add one or more expressions to exclude some file paths when searching for files. This can be achieve with the [+] button. These filters can use wildcard or regular expression syntax. Note that all file paths are described with '/' separator whatever the platform or system locale is. It is possible to remove filters by selecting the appropriate rows and then clicking on [—].
5. Select files that you want to process in the found files list. Unmarked files are not going to be read or touched. The content of this list is updated whenever you change search options. You can filter the content of this list through the file path filter above the list view. Click the column header to sort that column.

The scan of the harddrive for searching files is a pretty long operation when running on a very large directory structure, however it is possible to stop it at any time by clicking on [X] or by pressing the **Escape** key.

The column labelled *R/W* indicates the file permissions for the reading or writing. If the reading is not allowed on a file, it cannot be selected and consequently it cannot be processed. In that case the permissions appear on a red background.

File path	Encoding	BOM	End-of-line	R/W	Size	Modified
lector/fil...tor/PFilePathFilterProxyModel.cpp	ISO-8859-1	None	Unix	R/-	14...es	18.09.10 13:41
lector/file...lector/PFilePathFilterProxyModel.h	Unknown	None	Undefined	-/W Read only		18.09.10 13:41
lector/fileselector/PFileSelectorFrm.cpp	Unknown	None	Undefined	-/-	3 KB	17.04.09 20:10
lector/fileselector/PFileSelectorFrm.h	Unknown	None	Unix	R/W	80...es	18.09.10 13:41
lector/fi...r/debug/moc_PFileSelectorFrm.cpp	ISO-8859-1	None	Unix	R/W	3 KB	17.04.09 20:02
lector/fileselector/main.cpp	Unknown	None	Undefined	-/-	18...es	16.04.09 07:56
lector/fi...ctor/qtc-gdbmacros/gdbmacros.cpp	ISO-8859-1	None	Unix	R/W	77 KB	17.04.09 20:02
lector/fileselector/ui_PFileSelectorFrm.h	Unknown	None	Unix	R/W	15 KB	18.09.10 13:41

6.2   Detecting or Selecting the Encoding of Files

Character set detection is the process of determining the character set, or encoding, of character data in an unknown format. This is, at best, an imprecise operation using statistics and heuristics. Because of this, detection works best if you supply at least a few hundred bytes of character data that's mostly in a single language. In some cases, the language can be determined along with the encoding.

Several different techniques are used for character set detection. For multi-byte encodings, the sequence of bytes is checked for legal patterns. The detected characters are also check against a list of frequently used characters in that encoding. For single byte encodings, the data is checked against a list of the most commonly occurring three letter groups for each language that can be written using that encoding. The detection process is configured to ignore html or xml style markup, which can interfere with the detection process by changing the statistics.

The *Found files* section shows the found files according to your current options for searching files. The column *Encoding* is the only one that is editable by the user. You can choose by hand the appropriate codec for each file with a click on the appropriate row, in the *Encoding* column. This will show up a drop down list of available codecs as shown below:

	File path	Encoding	BOM	End-of-line	R/W	Size	Modified	Created
8	/home/common/dev...index-local.html	UTF-8	None	Unix	R/W	1 KB	21.07.09 18:08	11.09.10 10:47
9	/home/common/de...mple/index.html	UTF-8	None	Unix	R/W	1 KB	23.06.10 08:07	11.09.10 10:47
10	/home/common/de...ample/style.css	ISO-8859-1	None	Unix	R/W	53...es	21.07.09 18:08	11.09.10 10:47
11	/home/common/de...tes/product.php	ISO-8859-1	ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)				10 08:07	11.09.10 10:47

The "best codec" for processing text encoding of each file is detected using statistics and heuristics, and it is selected by default. Please note that these heuristics are only reliable for detecting Unicode charsets. For other encodings it will only give you some suggestions. There might are many "acceptable codecs" and they are all marked with a blue light on the left side of the drop down list. This is shown below:

Found files

Filter components

☐ Case sensitive filter   ☑ Case sensitive sort

	File path			
98	/home/philippe/dev/...in.content.html.php	ISO-2022-JP (JIS7)		0 AM
99	/home/philippe/dev/...t/admin.content.php	ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)		0 AM
100	/home/philippe/dev/..._content/config.xml	ISO-8859-10 (iso-ir-157, latin6, ISO-8859-10:1992, csISOLatin6)		0 AM
101	/home/philippe/dev/...content/content.xml	ISO-8859-13		0 AM
102	/home/philippe/dev/d...ntent/controller.php	ISO-8859-14 (iso-ir-199, latin8, iso-celtic)		0 AM

ISO-8859-15 (latin9)
ISO-8859-16 (iso-ir-226, latin10)
ISO-8859-2 (latin2, iso-ir-101, csISOLatin2)
ISO-8859-3 (latin3, iso-ir-109, csISOLatin3)
ISO-8859-4 (latin4, iso-ir-110, csISOLatin4)
ISO-8859-5 (cyrillic, iso-ir-144, csISOLatinCyrillic)

ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)

If the codec name is set to ***Unknown*** for a given row, that means that no codec seems to be consistent with the associated file content. When one or more codecs are detected as acceptable, the preferred text encoding is selected by default when it is available. The list of acceptable codecs is automatically determined whenever a new file is displayed in the list but the list is not updated when files are changed on your harddrive.

For refreshing the encoding detection, click on 🔍. This will detect the acceptable codecs for all found files listed. The detection process will attempt to identify the charset that best matches the characteristics of the byte data, but the process is partly statistical in nature, and the results can not be guaranteed to always be correct. For best accuracy in charset detection, the input data should be primarily in a single language, and a minimum of a few hundred bytes worth of plain text in the language are needed. The detection process will attempt to ignore html or xml style markup that could otherwise obscure the content. The charset detection is a pretty long operation when running on thousands of files, however it is possible to stop it at any time by clicking on ⊗ or by pressing the ***Escape*** key. If you need detecting the encoding on a few files only, you might use the command named ***Detect files encoding for selection*** in order to avoid performing the detection on thousands of files.

In order to increase performances of encoding detection as well as end-of-lines detection, all data are stored in a cache. That means that once a the text encoding (or line ending) has been detected, the detection does not occur anymore. These cached data are automatically refreshed whenever the file is modified. However, in case you really want to forcing a new detection, you can clear at any time the cached content with the command ***Clear current cached files encoding cache***, located in the context menu of the ***Found files*** table view. The number of files for which the se encoding information are cached can be set in the [preferences](#) panel.
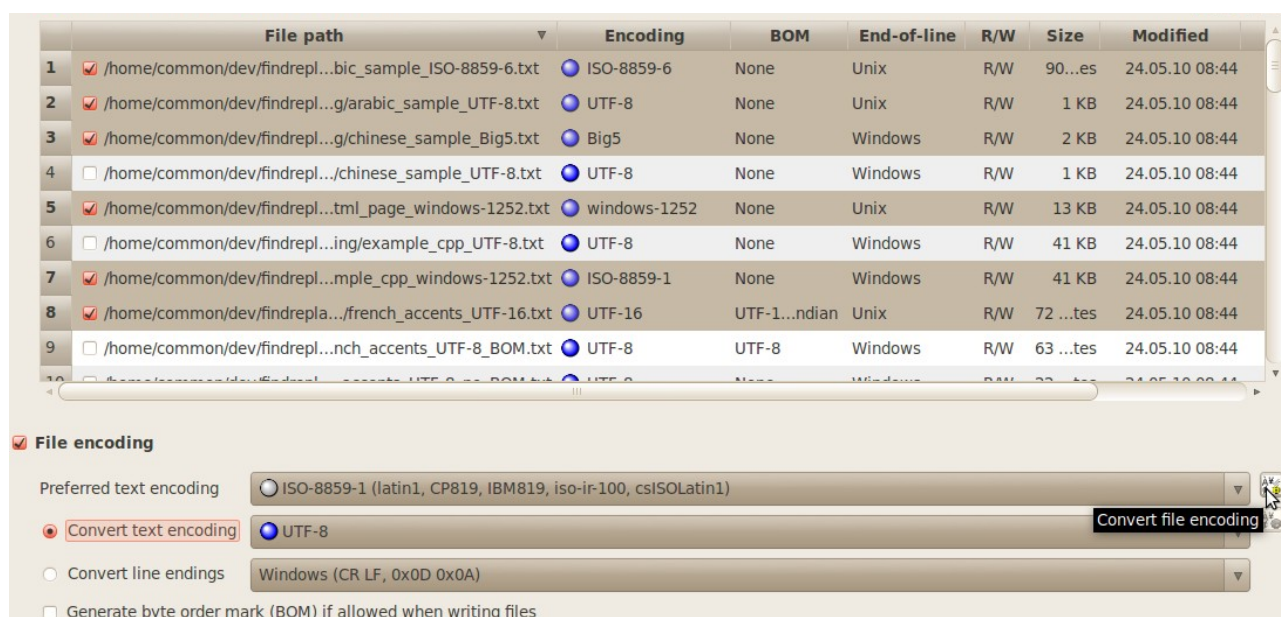
If you need to look at the decoded content of a file, right-click on a file entry, then click on ***Open file in test preview***. This will allow you to play with the codec used to decode the file.

Note: Character set detection is at best an imprecise operation. The text-encoding detection uses heuristics that do not always provide accurate results except for all Unicode encodings (i.e.: UTF-8, UTF-16, UTF-32).

6.3 *Converting the Encoding of Files*

For converting the text encoding of a given set of files, follow these steps:

1. Select the files you want to convert with a check mark in the ***File path*** column, within the ***Found files*** section (see screenshot below);

		File path	▽	Encoding	BOM	End-of-line	R/W	Size	Modified
1	☑	/home/common/dev/findrepl...bic_sample_ISO-8859-6.txt	● ISO-8859-6	None	Unix	R/W	90...es	24.05.10 08:44
2	☑	/home/common/dev/findrepl...g/arabic_sample_UTF-8.txt	● UTF-8	None	Unix	R/W	1 KB	24.05.10 08:44
3	☑	/home/common/dev/findrepl...g/chinese_sample_Big5.txt	● Big5	None	Windows	R/W	2 KB	24.05.10 08:44
4	☐	/home/common/dev/findrepl.../chinese_sample_UTF-8.txt	● UTF-8	None	Windows	R/W	1 KB	24.05.10 08:44
5	☑	/home/common/dev/findrepl...tml_page_windows-1252.txt	● windows-1252	None	Unix	R/W	13 KB	24.05.10 08:44
6	☐	/home/common/dev/findrepl...ing/example_cpp_UTF-8.txt	● UTF-8	None	Windows	R/W	41 KB	24.05.10 08:44
7	☑	/home/common/dev/findrepl...mple_cpp_windows-1252.txt	● ISO-8859-1	None	Windows	R/W	41 KB	24.05.10 08:44
8	☑	/home/common/dev/findrepla.../french_accents_UTF-16.txt	● UTF-16	UTF-1...ndian	Unix	R/W	72 ...tes	24.05.10 08:44
9	☐	/home/common/dev/findrepl...nch_accents_UTF-8_BOM.txt	● UTF-8	UTF-8	Windows	R/W	63 ...tes	24.05.10 08:44

☑ **File encoding**

Preferred text encoding   ● ISO-8859-1 (latin1, CP819, IBM819, iso-ir-100, csISOLatin1)   ▽   [Convert file encoding]

◉ [Convert text encoding]   ● UTF-8

◯ Convert line endings   Windows (CR LF, 0x0D 0x0A)   ▽

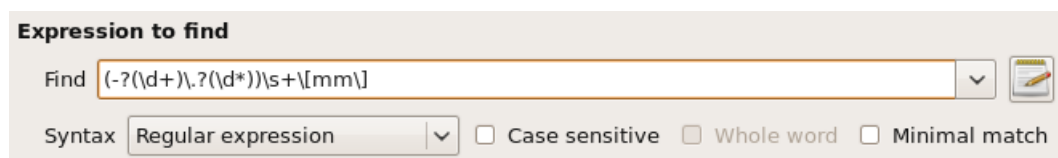☐ Generate byte order mark (BOM) if allowed when writing files

2. Select the current text encoding of these files if the auto-detected encodings are not accurate;
3. Select the target encoding for your set of files;
4. Optionally you can select the **Generate Byte Order Mark (BOM)** check box. This will insert the BOM (Byte Order Mark) at the beginning of the file when it is written. This option only apply to Unicode text encoding: UTF-8, UTF-16 and UTF-32. Note that this option may interfere with the target encoding. For instance, if you choose an Unicode encoding that does not allow the BOM, it will turn your target encoding to the closest Unicode encoding that allows it.
5. Optionally you might schedule a backup of modified files;

6. Click on the button 🔧 to start the encoding conversion. If you want to stop the conversion process, click on the button 🔧.

6.4 Searching for an Expression in Files

The **Find & Replace Editor** pane allows you to define what you want to search or replace.

The **Expression to find** section allows you to setup an expression to search for:

1. Enter the expression to search into the **Find** text field. Alternatively you might use the 📝 button to edit your expression with the **_Regular Expression Editor_**.
2. You can choose the way your expression must be interpreted through the **Syntax** drop down list:
   ○ Simple text or fixed string: means that the pattern to be matched is interpreted as a plain string

   ○ Wildcard: is similar to the functionality found in command shells
   ○ Wildcard Unix: This is similar to Wildcard but with the behavior of a Unix shell. The wildcard characters can be escaped with the character "".
   ○ Regular expression: is a pattern for matching substrings in a text
3. Select the options that apply when matching against your expression:

   The 'Minimal match' match option is only available when 'Regular expression'
   syntax is set. This turns the quantifiers in non-greedy mode.

4. Select the files you want to scan with a check mark in the **File path** column;
5. Select the current text encoding of these files if the auto-detected encodings are not accurate;
6. Optionally you can test your expression with the **_Find & Replace Preview_**;

7. Click on the 🔍 button to start searching your expression into the selected files. If you need to stop the search, click again on the same button which has been morphed into ❌ once the search has been started.

Note: For performance reasons it is not possible to search for an expression longer than 50'000'000 characters. By default this limit is fixed to 5'000'000 to increase search speed. This can be changed through the preferences panel.

6.5 Replacing a Given Expression in Files

The *Find & Replace Editor* pane allows you to define what you want to search or replace.

1. Setup the expression you want to search;

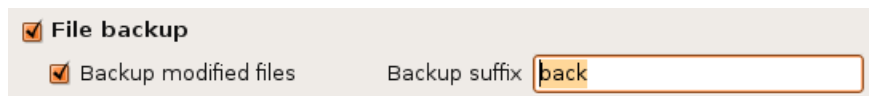2. Type your replacement pattern for matched occurrences of your expression in the *Replace with* text field:

☑ **Replacement expression**

Replace with | a replacement text | ▾

3. Optionally you might schedule a backup of modified files;

4. Click on the 🖉 button to start searching your expression into the selected files. If you need to stop the replacement, click again on the same button which has been morphed into ⊗ once the replacement has been started.

6.6 Doing Backups of Files before Modification

The *File backup* section allows you to create a backup for each modified files.

☑ **File backup**
    ☑ Backup modified files   Backup suffix | back

1. Mark the backup check box as shown above;

2. Type a suffix for your backup files. If a file with the same name already exists, it will be replaced. So you should consider to update the file suffix between each manipulation of the original file.

6.7 Using the Find & Replace Preview

When editing an expression to search for, it is convenient to match it against real data. In order to achieve this, let's take a tour of the *Find & Replace Preview* window.

With the preview window you can visualize in one glance what are the texts who match a given expression, what will be the replacement text and even mix both views. It gives you an immediate feed-back on what is going to change which makes easy to understand the impact of your replacement within a file without actually modifying its content. Such a realtime visualization is especially useful whenever you want to build a complex expression to search for. The preview can also be used as a multi-document editor.

Here are the key features of this tool:

- check the impact of a given text encoding when applied to a file content;

- edit a text sample against which you want to match an expression to find;

- preview matched occurrences of an expression to find inside a given text sample;

- preview resulting content of a text sample after the replacement of all occurrences of your expression with your replacement pattern;
- preview both found expressions and processed replacements inside a text sample;

- navigate through found occurrences of a specific expression or replacement pattern;

- preview HTML documents.

6.7.1 Previewing as Plain Text

You can open files in the preview by clicking on [icon]. The button [icon] allows saving the content of the preview. The name displayed in the tab will be used as the file name for saving. If the label of the tab is not a valid file path you will have to select a destination file. You can force to select a destination file path by using the dropdown arrow beside the button, and select **Save as...**.

The dropdown button [icon] let you start a new preview document, or use the contents of the active preview as source text for a new preview ([icon]).

The dropdown button [icon] let you clear the contents of the active preview, close the current preview tab, or close all tabs.

It is possible to preselect the preferred codec for the preview of upcoming files. In fact, the active codec of the preview is used as preferred codec for the next opened file. It is also possible to reload the content of the file in the preview with the button [icon]. Thus the selected codec will be used to read the file.

It is possible to copy the file path of the active preview to the clipboard by right-clicking on the tab bar. In the same way it is also possible to open the active document within the default editor associated to this type of file.

Note that all line breaks in the preview are internally represented by a Line Feed character (LF, U+00A). This is always true, whatever the original end-of-line used in the displayed file. If you want to search for a multi-line expression with another style of line break, we strongly advise you to use a regular expression with appropriate *\s+* sequences in order to match any kind of end-of-line.

The following screenshots illustrate some of the capabilities described above.
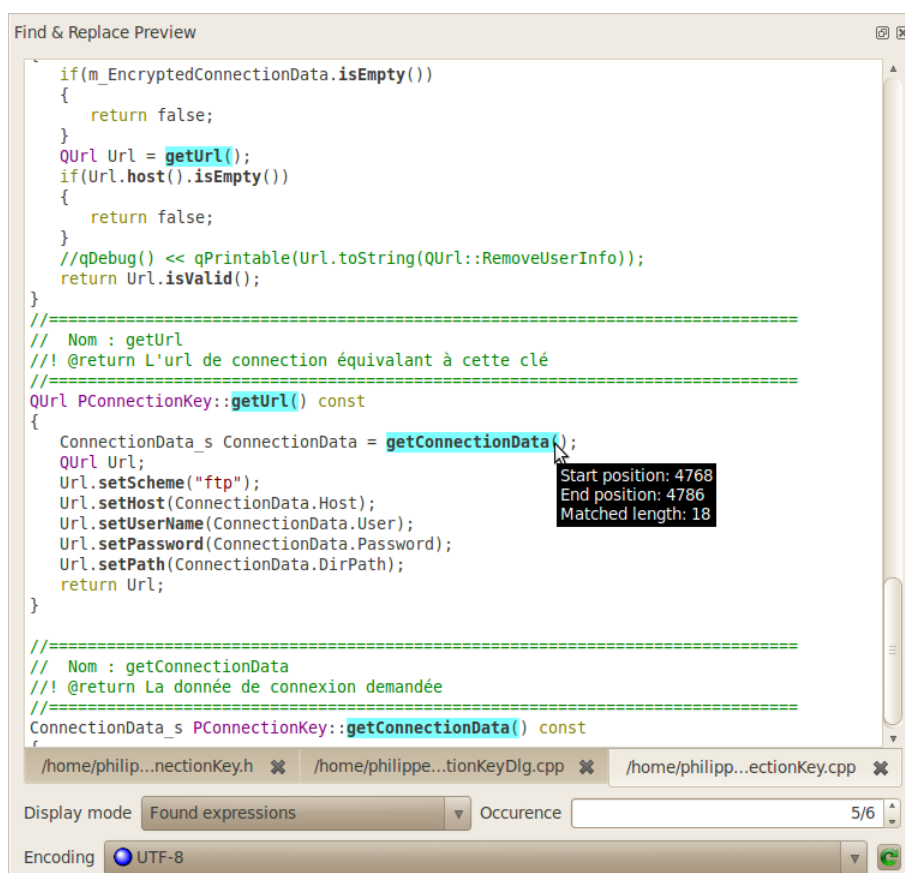
Highlight of matched expressions within a file. The tool-tip shows information about occurrence location.

```
        }
        QUrl Url = url();
        if(Url.host().isEmpty())
        {
            return false;
        }
        //qDebug() << qPrintable(Url.toString(QUrl::RemoveUserInfo));
        return Url.isValid();
    }
    //==========================================================================
    //  Nom : getUrl
    //! @return L'url de connection équivalant à cette clé
    //==========================================================================
    QUrl PConnectionKey::url() const
    {
        ConnectionData_s ConnectionData = connectionData();
        QUrl Url;
        Url.setScheme("ftp");
        Url.setHost(ConnectionData.Host);
        Url.setUserName(ConnectionData.User);
        Url.setPassword(ConnectionData.Password);
        Url.setPath(ConnectionData.DirPath);
        return Url;
    }


    //==========================================================================
    //  Nom : getConnectionData
    //! @return La donnée de connexion demandée
    //==========================================================================
    ConnectionData_s PConnectionKey::connectionData() const
    {
```
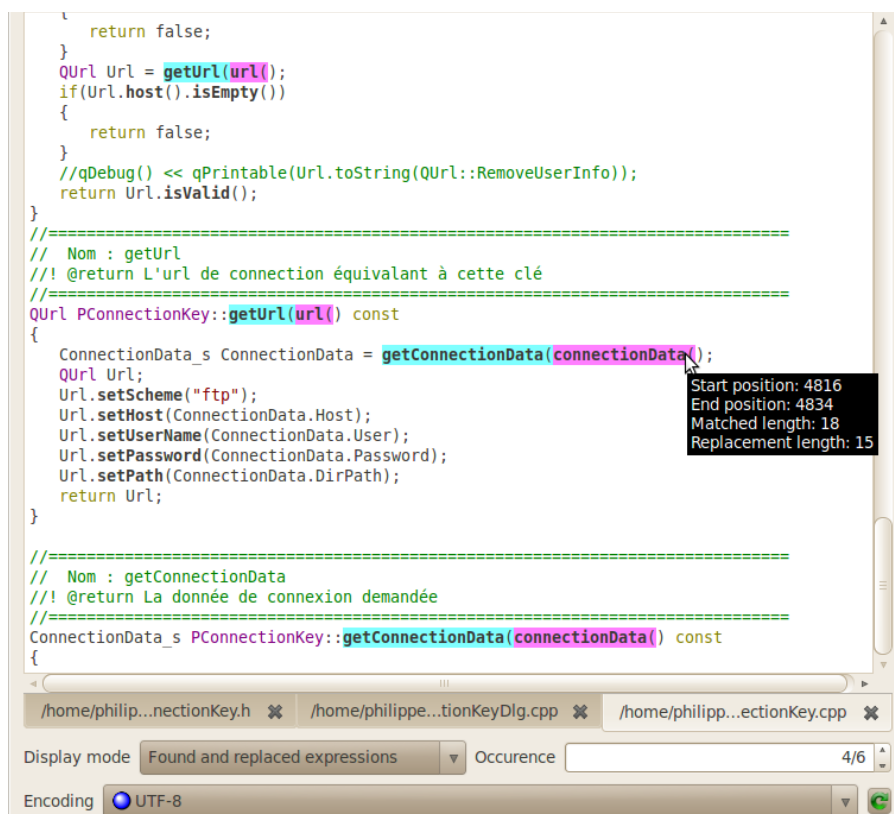
Editable source text
Found expressions
Replaced expressions
Found and replaced expressions

/home/philip...   ...KeyDlg.cpp ✖   /home/philipp...ectionKey.cpp ✖

Display mode   Occurence   2/6

Encoding ● ...

Highlight of replaced expressions within a file.

```
    {
        return false;
    }
    QUrl Url = getUrl(url());
    if(Url.host().isEmpty())
    {
        return false;
    }
    //qDebug() << qPrintable(Url.toString(QUrl::RemoveUserInfo));
    return Url.isValid();
}
//==========================================================================
//  Nom : getUrl
//! @return L'url de connection équivalant à cette clé
//==========================================================================
QUrl PConnectionKey::getUrl(url()) const
{
    ConnectionData_s ConnectionData = getConnectionData(connectionData());
    QUrl Url;                               Start position: 4816
    Url.setScheme("ftp");                   End position: 4834
    Url.setHost(ConnectionData.Host);       Matched length: 18
    Url.setUserName(ConnectionData.User);   Replacement length: 15
    Url.setPassword(ConnectionData.Password);
    Url.setPath(ConnectionData.DirPath);
    return Url;
}


//==========================================================================
//  Nom : getConnectionData
//! @return La donnée de connexion demandée
//==========================================================================
ConnectionData_s PConnectionKey::getConnectionData(connectionData()) const
{
```

/home/philip...nectionKey.h ✖   /home/philippe...tionKeyDlg.cpp ✖   /home/philipp...ectionKey.cpp ✖

Display mode Found and replaced expressions ▼ Occurence   4/6

Encoding ● UTF-8   ▼ ⟳

Highlight of both found and replaced expressions within a file. The tool-tip shows information about replacement location.

Note: For performance reasons the *Find & Replace Preview* has a content limit of 100'000'000 characters. By default this limit is fixed to 5'000'000 to increase search speed. This can be changed through the preferences panel.

6.7.2 Previewing as HTML

In addition to the plain text preview, you can activate the **HTML Viewer** through the 🔵 button. That will enable you to preview HTML documents, with either their original or altered content, without having to save them.
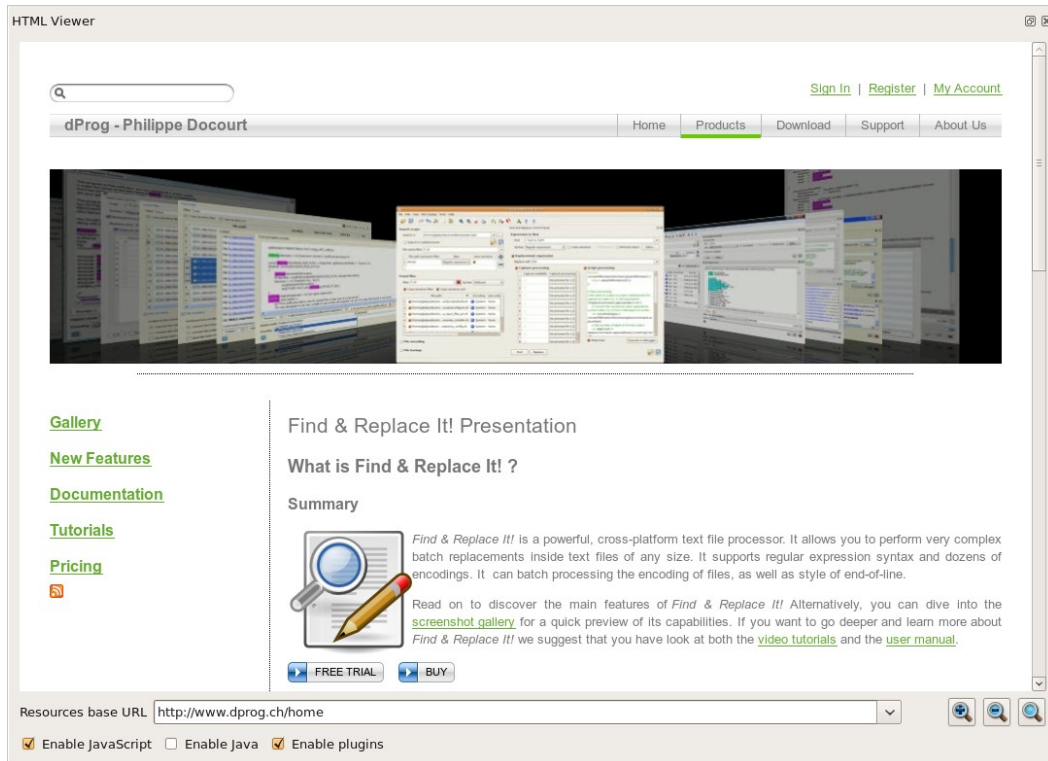
The HTML preview in action with our own website.

The viewer is directly synchronized with the current content of plain text preview. That means the **Display mode** will also apply on the HTML content.

Because the viewer cannot resolve relative links in the HTML document from the content of the preview, you might need to enter an appropriate URL. This URL is used to process all resources referenced by relative links within the document (i.e.: CSS, images, scripts, etc.). This is not required when all resources are given with absolute path.

The **HTML Viewer** provides rendering of HyperText Markup Language (HTML), Extensible HyperText Markup Language (XHTML) and Scalable Vector Graphics (SVG) documents, styled using Cascading Style Sheets (CSS) and scripted with JavaScript. Some common plugins are also supported through the Netscape Plugin API, provided you have appropriate binary files for those plugins installed on your computer. The following locations are searched for plugins:

Linux/Unix

- .mozilla/plugins   in the user's home directory
- .netscape/plugins   in the user's home directory
- System locations, such as
    - /usr/lib/browser/plugins
    - /usr/local/lib/mozilla/plugins
    - /usr/lib/firefox/plugins
    - /usr/lib64/browser-plugins
    - /usr/lib/browser-plugins
    - /usr/lib/mozilla/plugins
    - /usr/local/netscape/plugins
    - /opt/mozilla/plugins

<div align="center">**Linux/Unix**</div>

- /opt/mozilla/lib/plugins
- /opt/netscape/plugins
- /opt/netscape/communicator/plugins
- /usr/lib/netscape/plugins
- /usr/lib/netscape/plugins-libc5
- /usr/lib/netscape/plugins-libc6
- /usr/lib64/netscape/plugins
- /usr/lib64/mozilla/plugins
- Locations specified by environment variables:
    - $MOZILLA_HOME/plugins
    - $MOZ_PLUGIN_PATH
   - $QTWEBKIT_PLUGIN_PATH

<div align="center">**Windows**</div>

- The user's   Application Data\Mozilla\plugins   directory
- Standard system locations of plugins for Quicktime, Flash, etc.

<div align="center">**Mac OS X**</div>

- Library/Internet Plug-Ins   in the user's home directory
- The system   /Library/Internet Plug-Ins   directory

6.8 *Using the Regular Expression Editor*

When you need to write a multi-line expression to search, the ***Regular Expression Editor*** is your best friend. The multi-line edition is shown bellow with a ***simple text*** expression:
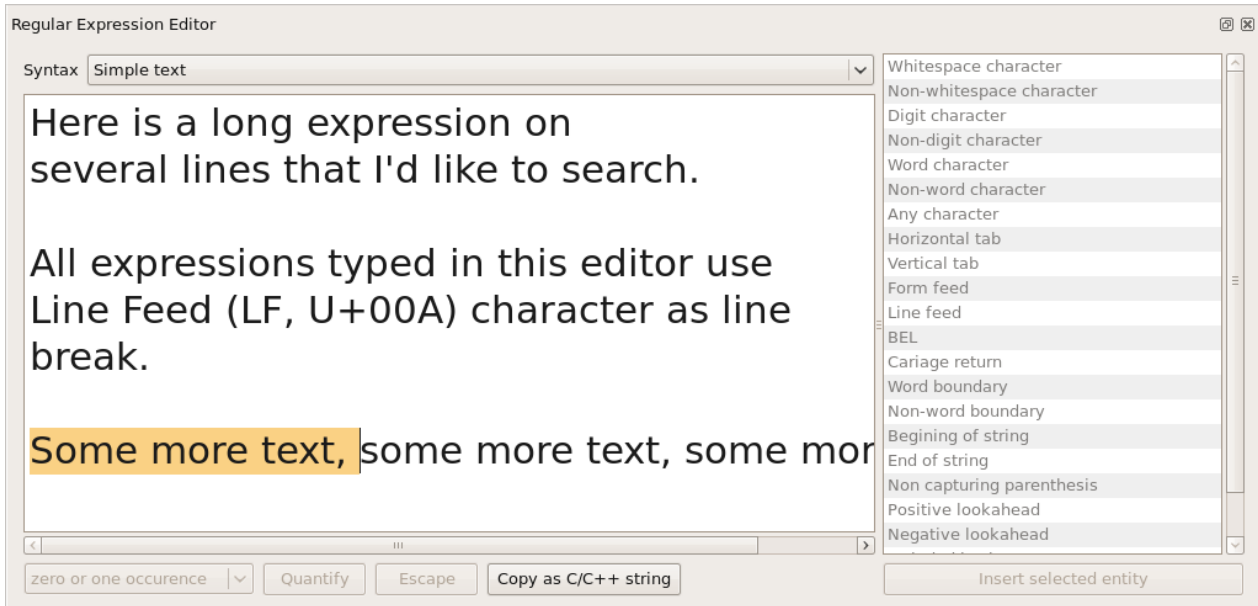

Note that all line break are internally represented by a Line Feed character (LF, U+00A). If you want to search for a multi-line expression with another style of line break, we strongly advise you to use a regular expression with appropriate '\s+' sequences in order to match any kind of end-of-line.

In addition, this editor simplifies the setup of regular expression. It provides tools to manage regular expression entities. On the right side of the text editor, there is a list of available regular expression entities (e.g.: special characters, grouping expression, etc.). If you leave the mouse over an item of this list, a explanatory tooltip will appear. This is show in the figure bellow:
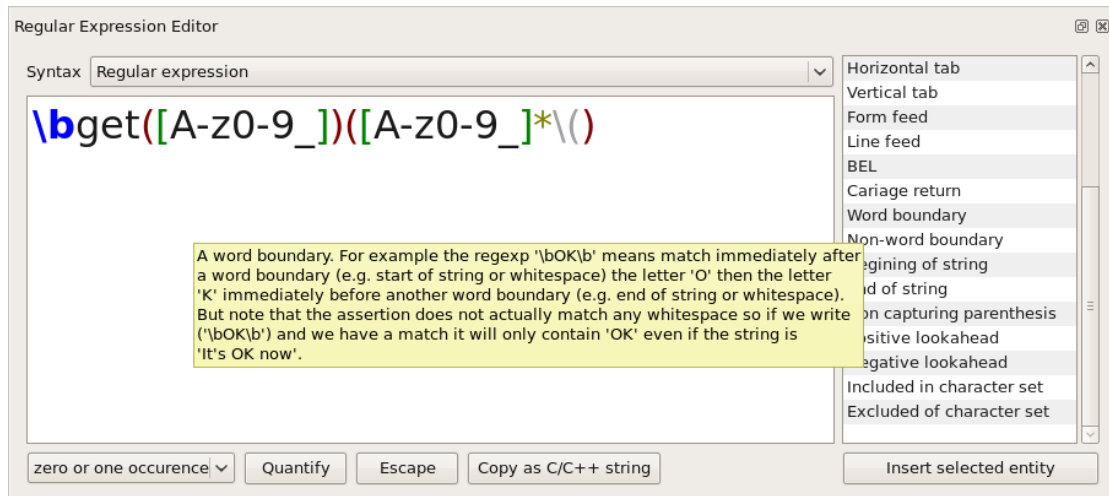
The Regular Expression Editor has some nice features like syntax highlighting and scope matching (e.g.: matching scope for '()', '[]', '{}').

The **Regular Expression Editor** offers an automatic syntax check for wildcard and regular expressions:
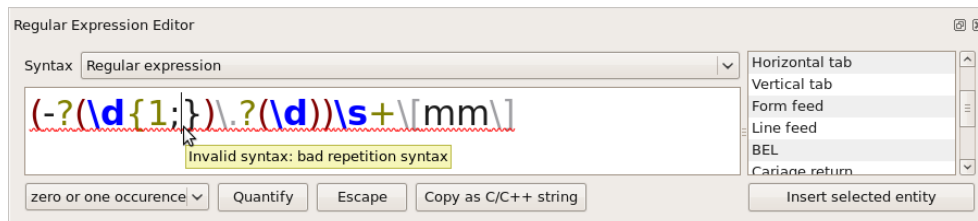
As soon as your wildcard or regular expression pattern becomes invalid, it is underlined. A tooltip provides a brief description of the syntax error detected.

6.9 Advanced Replacements

Advanced replacement covers three main features that makes *Find & Replace It!* really powerful:

- Injecting a fragment of the matched expression into the replacement text;
- Transforming a fragment of the matched expression before injecting it into the replacement text;
- Transforming any text that is going to be replaced by using a JavaScript interface.

Each of these points is described in the following chapters.

6.9.1 Using of Captured Texts within Replaced Texts

This feature requires regular expression syntax for the expression to find; furthermore, you should be familiar with captures within regular expressions. To learn more about these notions we recommend to read the Regular Expressions chapter. If you are familiar with regexp, read on the following example.

Whenever you capture some text fragments with an expression to find,

you can inject these captured fragments into your replacement pattern. This is done with a *%1*, *%2*, …, *%9* pattern. Where the number that follows the percent sign is the capture index. *%0* is a special, implicit capture that includes the full matched expression.

Replace with %1 replacement text %2 ▼

Let's imagine we have a CSV file containing contacts like in the following snippet:

```
First Name: John; Family Name: Smith; Phone: ...
First Name: Mike; Family Name: Dupont; Phone: ...
```

We want to swap the first two columns. Here we have to capture two variable expressions (first name and family name) and move them around. Here is an easy way to do it.

Find:

(First Name: [^;]+); (Family Name: [^;]+)

The parentheses in expression above will capture two fragments of every matched occurrences in the CSV file.

Replace with:

%2; %1

The replacement pattern above is a dynamic text that varies for every matched occurrences. In fact *%1* will be replaced by the content matched by the first parentheses scope. Idem with *%2* and the second parentheses scope.

If you use *%n* where n is greater than the number of captures, then *@{__INJECTION__}#n* will be inserted in the replaced text. If you need to replace some text with a literal *%n* in a situation where you captured some texts with your regexp, you'll need escaping the *%n* sequence. This can easily be done by adding a *%* sign in front of the sequence. Note that the escapement only applies in front of a digit (i.e.: 0 to 9). Everywhere else the escapement is not processed. Here are some examples:

Text to process	Find	Replace with	Result
ab cd	(\w+)	%1	ab cd
56 78	(\d+)	%1%2	56@{__INJECTION__}#2 78@{__INJECTION__}#2
56 78	(\d+)	%1%%2	56%2 78%2
56	\d+	%%1	%1
56	(\d+)	%%1	%1
56 78	(\d+)	%%%1	%56 %78
56 78	(\d+)	%%%%1	%%1 %%1
56 78	(\d+)	%%%%2	%%2 %%2
56 78	(\d+)	%%%%%1	%%56 %%78
56 78	(\d+)	%%%%%2	%%@{__INJECTION__}#2 %%@{__INJECTION__}#2
ab	(\w+)	%z %1	%z ab
ab	(\w+)	%%z %%1	%%z %1
ab	(\w+)	%%%z %%%1	%%%z %ab

In the examples above, the pattern *\w+* matches any sequence of one or more letters or digits. The same applies to digits only with the pattern \d+. These examples assume that the regexp is not set to "minimal match". Please read the Regular Expressions chapter for more details on the regexp syntax.

6.9.2 Processing the Captured Texts with Built-In Functions

This feature requires regular expression syntax for the expression to find; furthermore, you should be familiar with captures within regular expressions. To learn more about these notions we recommend to read the Regular expressions chapter. If you are familiar with regexp, read on the following example.

As shown in the previous chapter, it is possible to inject captured texts into your final replacement string, through a special syntax. It is also possible to apply an additional processing to captured strings before injecting them as a replacement expression. This can be handle with the **Capture processing** section:

Capture #1 and #2 are available but not capture #3. A distinct process has been attached to each capture, however it is not compulsory.

The left column is not editable. The check mark is toggled depending on the presence of captures and placeholders respectively within the expression to find and within the replacement pattern. The right column let you choose a transformation to apply to the capture, before injecting it as the replacement text at its placeholder location.

Let's imagine that we want to upper case the first letter that follows a ':' sign inside a file. A tedious solution might be to replace all *: a* with *: A*, *: b* with *: B* and so on. This will take some time. And then, what happens with accentuated letters or oriental characters ? What happens if a tab sometimes replaces the whitespace after the ':' sign? What if there no whitespace at all or many white-spaces due to a typing mistake? This solution is definitively inappropriate.

Here is a better way to handle this task:

Find:

```
:\s*(\w)
```

The expression above will match all ':' followed by any number of whitespace characters (including tabs and line breaks) and at at least a word character.

In the **Capture processing** section, select **To upper case** as process for the first capture. This will transform to upper case the content matched by the first pair of parentheses, before injecting it as *%1* in the replacement pattern.

Finally we replace with:

```
: %1
```

If the built-in capture processes are not sufficient, you might try the scripting interface.

6.9.3 Processing the Replaced Texts by Script

Find & Replace It! has the ability to dynamically transform the texts to be replaced. This involves mixing all the classical possibilities of regular expressions with JavaScript programming. It is possible to look for very complex patterns with regular expressions. However a simple replacement by a static text is most of the time insufficient. Hopefully, *Find & Replace It!* provides a JavaScript-like interface to customize replacements on the fly. The scripting feature gives all the power of JavaScript to format whatever you need and the regular expressions simply select where to apply the script. This feature especially useful whenever you need some logic to interpret an expression and generate a replacement pattern based on it (e.g. find all numbers in a text and divide them by a given factor to convert units).

One of the most powerful feature of *Find & Replace It!* is its ability to dynamically transform the texts to be replaced. This involves mixing all the classical possibilities of regular expressions with JavaScript programming. It is possible to look for very complex patterns with regular expressions. However a simple replacement by a static text is most of the time insufficient. Therefore, the scripting feature gives us all the power of JavaScript to format whatever we need and the regular expressions simply tell us where to apply the script.

To carry out any of the mentioned examples above you only have to write an appropriate script in the script editor. Your script will be called at three different occasions. All of them can be accumulated as desired:

- Once with the original replacement pattern and the full matched expression;

- Once for each captured text in your expression if the capture process has been set to **Apply script**;

- Once at the very end of the process, after all capture processing, with the resulting replacement pattern.
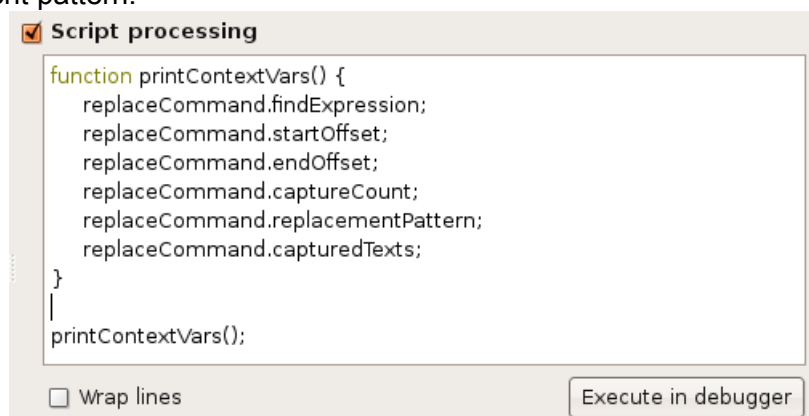
A dummy script outputting invariable properties from the scripting context.

The scripting interface provides a simple way to access the context of the current match as well as the replacement pattern through the global `replaceCommand` object. The table below summarizes the context information made available to the script:

Properties of replaceCommand	Object or Data Type	Description
Invariable properties: these variables do not change during the process of a given matched expression		
`findExpression`	RegExp	Matched regular expression.
`startOffset`	Number	Starting offset of matched expression in the full text.
`endOffset`	Number	Ending offset of matched expression in the full text.
`captureCount`	Number	Number of captures contained in the expression to find.
`capturedTexts`	Array	Array of captured strings within the full matched expression.

Variable properties: these variables vary depending on the capture for which the script is called		
captureIndex	Number	Index of the current capture for wich the script has been called. This index is included between 0 and 'captureCount + 1'.
captureText	String	Captured text at current capture index. This property returns the full matched expression when captureIndex is zero and returns an empty string when captureIndex equals 'captureCount + 1'. Otherwise it returns the captured string at the given index starting from 1.
replacementLength	Number	Length of the current replacement text. This length takes in account already replaced placeholders (i.e. *%i*).
replacementText	String	Current replacement text. This text contains the content of already replaced placeholders (i.e. *%i*). All placeholders that have not been replaced at the time of accessing this property (i.e.: i>***aptureIndex***) are internally represented by *@{__INJECTION__}#i*. All escaped sequences of *%* are already treated in the text returned by this property.

The scripting interface also provides a simple way to access some properties of the file being parsed. You can read these properties through the global ***currentFile*** object. The table below summarizes the context information made available to the script:

| Properties of currentFile	Object or Data Type	Description
Invariable properties: these variables do not change during the process of a given matched expression		
filePath	String	Full file path (absolute) including the name of the file.
dirPath	String	Absolute directory path where the file is located.
fileName	String	Full file name, including the extension.

Let's imagine we have a text file containing numerous numerical values. All of these values represent a length given in millimeters. We would like to convert all these distances from millimeters to inches. Sounds tricky to you? Script processing enables you to handle that task like any other replacement task. First we setup an expression that matches all numbers (integers and floating point):

> *(-?(\d+)\.?(\d*))*

As a replacement we simply inject the full captured number:

> *%1*

In order to apply some script on the replacement pattern activate capture processing (i. e. check mark), set process to ***Apply script***, then activate the script processing editor option as shown below:
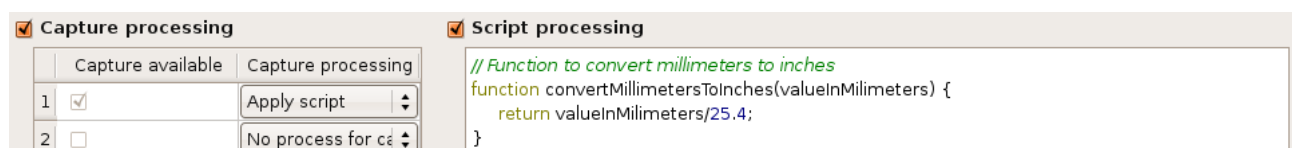
Finally copy and paste the following script within the script editor:

```
// Function to convert millimeters to inches
function convertMillimetersToInches(valueInMilimeters) {
      return valueInMilimeters/25.4;
}


// We want to output a custom replacement for capture at index 1
if(replaceCommand.captureIndex==1) {
      // Convert the numerical value captured at current
      // index (i.e 1) from millimeters to inches
      var convertedValue =
            convertMillimetersToInches(replaceCommand.captureText);
      // Get number of digits to format output
      var digitCount =
            replaceCommand.capturedTexts[2].toString().length;
      // Take in account that the converted value is about ten times
      // smaller, therefore we subtract one digit to represent it
      // with same accuracy
      digitCount -= 1;
      // Get the number of captured decimals (original decimal count)
      digitCount += replaceCommand.capturedTexts[3].toString().length;
      // Ouput the formatted value with an equivalent number of digits
      convertedValue.toPrecision(digitCount);
} else {
      // For the other indexes (i.e. 0, 2, 3, 4) we simply output
      // the current replacement text
      replaceCommand.replacementText;
}
```

That's it! As an alternative we might remove the outer parentheses, then use *%0* as replacement pattern instead of *%1* and adjust the script accordingly.

6.10 Using the Output Window

The **Output** window is a multi-document preview for find and replace reports. Every time you search for an expression within files, a report will be outputted to the active console tab. The report includes links to files that contain searched/replaced expressions. A simple click on this link will open the file within **_Find & Replace Preview_**. A double-click on this link will open the file with a suitable application.
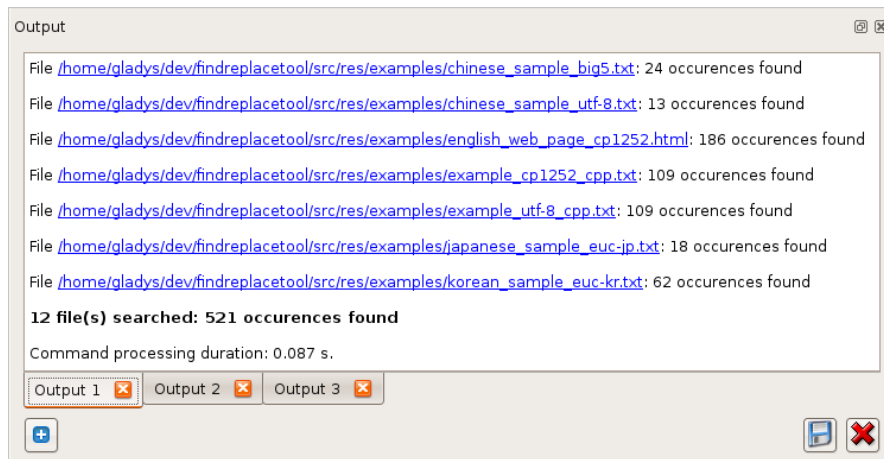
The console showing a report for found expressions. The report gives you some statistics.

6.11 Editing the Preferences

The preferences window lets you customize some settings. This window is accessible via the menu **_Edit/Preferences..._**. On Mac OS X, this entry is located in the application menu as usual.

The available preferences are:

- **_General_**

  - **_Language_**: This field indicates the current language used for the user interface. You must restart the application for this change taking effect.

  - **_Max length of matched expressions_**: This field indicates the maximum number of chars that can be matched by a regexp. It is not possible to find or replace a string that is longer than this value. For performance reasons, it is strongly recommended not to increase too much this value.

  - **_Max size of files encoding cache_**: This field indicates the maximum number of files for which the detected encoding data are cached. The cache dramatically speeds up the encoding detection process. However, a very big cache can slow down the startup of the application, due to the time for loading cache. This time can usually be neglected in comparison to the performance increase during encoding detection.

  - **_Used preferred text encoding if no acceptable codec is detected_**: Checking this box will force the preferred encoding whenever no acceptable encoding can be automatically proposed. This will force also the preferred codec for binary files.

- **_Find & Replace preview_**

  - **_Max size of preview_**: This field indicates the maximum number of chars that can be loaded in the preview window. If you try to load a file which is longer than this value, the end of the file will be ignored. For performance reasons, it is strongly recommended not to increase too much this value.

- ○ ***Color of found expressions***: This editor shows the background color used to highlight found expressions.

- ○ ***Color of replacement text***: This editor shows the background color used to highlight replaced text.

- ○ ***Wrap lines***: Checking this box will cause words to be wrapped at the right edge of the text area. Wrapping occurs at whitespace, keeping whole words intact.

- ○ ***Activate syntax highlighting***: Checking this box will activate syntax highlighting in the preview area. Syntax highlighting is only available for JavaScript, C/C++ and XML/HTML files.
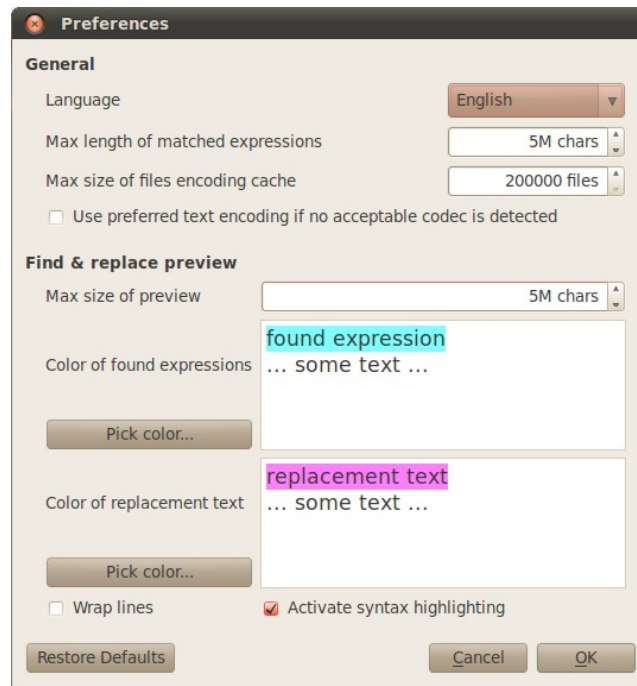
The preferences window with default settings

In order to restore the default settings, simply click on ***Restore Defaults***.

6.12 Using the Command Line

It is not possible to do replacements directly from the command line yet. However, the command line offers a few interesting features:

Command	Action/Effect
./FindReplaceIt ***-locale*** <locale_code>	Changes the locale used to <local_code>. E.g.: *FindReplaceIt -locale fr_CH* activates the Swiss French locale.
./FindReplaceIt ***-reset-settings***	Resets all settings: clears pre-loaded content, resets font settings, resets window positions, etc. The application is not launched.
./FindReplaceIt ***-trace*** *[<path/to/trace/file>.xml]*	Enables tracing of application's activity. This enables advanced logging for debug purpose in case of problems. This might be useful for reporting a crash to our support team. Without the optional argument that specify a custom path for the trace file, this command will generate by default a trace

	file named *trace.xml* alongside the application executable file.
./FindReplaceIt *-version*	Prints the full name and version of the application. The application is not launched.
./FindReplaceIt **<***path/to/fri/file***>.fri**	Opens the given file.

6.13 Using Find & Replace It! documents

Working with documents allows you to save and recover your work. However, it is possible to work with or without an active document within *Find & Replace It!*. By default there is no document created when the application is loaded. In other words there is no active document set when the application starts.

Here are the three ways of starting to work with an active document:

- Using the **File/New** menu : starts a new document and resets the user interface

- Using the **File/Open...** menu : the opened document becomes the current one

- Using the **File/Save** menu : the current content of the user interface is saved, and the freshly saved document becomes the new current document

Having an active document has the following effects:

- Every time you try to quit the application, you'll be asked to save your modifications if any;

- Every time you open, or close a document, you'll be asked to save your modifications if any;

- The name of the current document is shown in the title bar of the main window;

- Using the **File/Save** menu will overwrite the document previously saved.

You might close the active document at any time by using the **File/Close** menu. This command will also reset the content of the user interface.

The documents generated by *Find & Replace It!* are suffixed by *.fri*. They are called **UI** files (i.e.: User Interface File) for *Find & Replace It!*.

All sections containing this button support persistent serialization. Therefore, you can save an expression alongside with its replacement pattern and script, as well as all components for searching and filtering files. Of course the twin button enables you to load a file previously saved.

As mentioned before, the **File** menu provides a way of saving and loading the full content of the interface. In fact, using the **File/Save** menu is almost equivalent to concatenate all files generated by all buttons of all sub-windows.

If you open a *.fri* file from a sub-window button (e.g.: **_Find & Replace Editor_**), only the content related to this sub-window will be loaded. Hence you can easily load only a part of a file saved from the **File/Save** menu. This also means that if you open a file generated from a sub-window inside another sub-window, nothing will be loaded. In the same way, opening a file generated from a specific sub-window with the the **File/Save** menu, only loads the content of the specific sub-window, leaving all other windows intact.

7   Tips and Tricks

7.1 Multi-document Tabs

Some components use tabs to display multiple documents. To manually open a new document/tab, use this button ⊕. To close a document/tab, click on the button ⊗ located on the corresponding tab.

7.2 Working with Text Areas

7.2.1 Navigating text

The content of any text area can navigated with the following key bindings:

Keypresses	Action
LeftArrow	Moves the cursor one character to the left.
Ctrl+LeftArrow	Moves the cursor one word to the left.
RightArrow	Moves the cursor one character to the right.
Ctrl+RightArrow	Moves the cursor one word to the right.
UpArrow	Moves the cursor one line up.
DownArrow	Moves the cursor one line down.
PageUp	Moves the cursor one page up.
PageDown	Moves the cursor one page down.
Home	Moves the cursor to the beginning of the line.
Ctrl+Home	Moves the cursor to the beginning of the text.
End	Moves the cursor to the end of the line.
Ctrl+End	Moves the cursor to the end of the text.
Ctrl+G	Moves the cursor to the beginning of a given line. This opens a popup for selecting the line number where one wants to jump to.

7.2.2 Editing Text

This is the list of key bindings which are implemented for editing:

Keypresses	Action
Backspace	Deletes the character to the left of the cursor.
Delete	Deletes the character to the right of the cursor.
Ctrl+C	Copy the selected text to the clipboard.
Ctrl+Insert	Copy the selected text to the clipboard.
Ctrl+K	Deletes to the end of the line.
Ctrl+V	Pastes the clipboard text into text edit.
Shift+Insert	Pastes the clipboard text into text edit.
Ctrl+X	Deletes the selected text and copies it to the clipboard.
Shift+Delete	Deletes the selected text and copies it to the clipboard.
Ctrl+Z	Undoes the last operation.

Keypresses	Action
Ctrl+Y	Redoes the last operation.
Alt+Wheel	Scrolls the page horizontally (the Wheel is the mouse wheel).
Ctrl++	Zooms in the text.
Ctrl--	Zooms in the text.
Tab (in some text areas)	Indents the current selection. Insert a tab if the the selection is empty.
Shift+Tab (in some text areas)	Unindents the current selection. Insert a tab if the the selection is empty.

To select (mark) text hold down the Shift key whilst pressing one of the movement keystrokes, for example, Shift+Right will select the character to the right, and Shift+Ctrl+Right will select the word to the right, etc.

7.2.3 Undo/Redo Changes

It is possible to undo and redo any change made in a text area when this area is editable. On the *Edit* tool bar or in the *Edit* menu, simply click on:

- Undo (Ctrl+Z)
- Redo (Ctrl+Y)

7.2.4 Changing Display Properties

The commands located in the *Text Display* menu let you change the appearance of the active text zone content:

- Zoom out (Ctrl++)
- Zoom in (Ctrl+-)
- Select a font

All these commands are available for all text area int the software but they only apply to the last area that has been activated. Therefore you might have to click somewhere inside a text area to get a result.

The default font and zoom can be restored with the *Text Display/Restore default font* command.

It is possible to print the content of the active text area through the *Text Display/Print...* command.

7.2.5 Searching for Text in Text Areas

It is possible to search for text within any text area of the graphical interface. Simply click on in the *Edit* tool bar or in the *Edit* menu. This will show up the *Find Text* window:
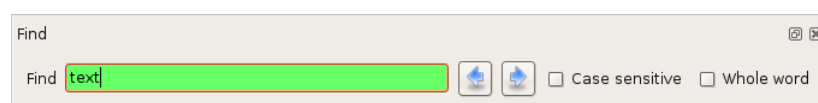
The search will occur in the last area that has been activated. The background is colorized in green when there is a match, in red when there is no match.

7.3 Using Logical Folders for Searching Files

Sometimes searching files in a folder given by its absolute path is not convenient. For instance it does not allow you to reuse your configuration (saved in a *.fri* file) on a different computer when files are not located in the same folder. Moreover, you might want to distribute *.fri* files without showing your file system structure to people who receive the *.fri* files.

In order to specify a folder to search for files without actually having to type its absolute path we provide three logical folders accessible through the following variables:

- **$(APP_DIR)**: This variable refers to the directory's path that contains the application executable. It varies depending on where the software has been installed.

- **$(CURRENT_DIR)**: This variable refers to the absolute path of the application's current directory.

- **$(HOME_DIR)**: This variable refers to absolute path of the user's home directory.

Here is an example of how to use the variable **$(HOME_DIR)**:
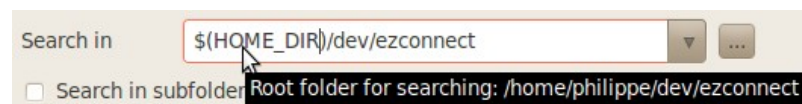
The tool tip shows the real file path obtained after evaluating the variable.

7.4 Multi File Selection in the Found Files List

To operate on many files at once it is possible to use the context menu on the **Found files** list. This menu let you act on the the current selection. As shown below, possible actions on selected files are: toggling check marks, selecting encoding, loading files in the preview. Applying an action on a selection of files. Note that **Open file in test preview** will open all selected files as distinct documents within **Find & Replace Preview**.

7.5 Getting Examples

There are some sample files shipped with **Find Replace It!**. Under Windows and Linux these files are located within the following directory:

```
<application install dir>/res/examples
```

On Mac OS X, these files are located in:

```
<application install dir>/Find & Replace It!.app/Contents/Resources/examples
```

The files suffixed with `fri` are **UI** files (i.e.: <u>User Interface File</u>) for *Find & Replace It!*. Such files contain stored user interface data and can be opened with 📂 buttons.

The files suffixed with `.txt` are sample data provided for convenience, in order to easily test the capabilities of *Find & Replace It!*.

Most examples provided alongside *Find & Replace It!* are available as a separate package which can be downloaded from <u>http://www.dprog.ch/home/download/</u> (see *Find & Replace It! Advanced Replacement Examples*).

7.6 *Debugging Script*

When willing to script some replacement texts it is convenient to debug the script. *Find & Replace It!* comes with an integrated debugger. To start the debugger, click on the **Execute in debugger** button located under the <u>script editor</u>. The debugger will show up:
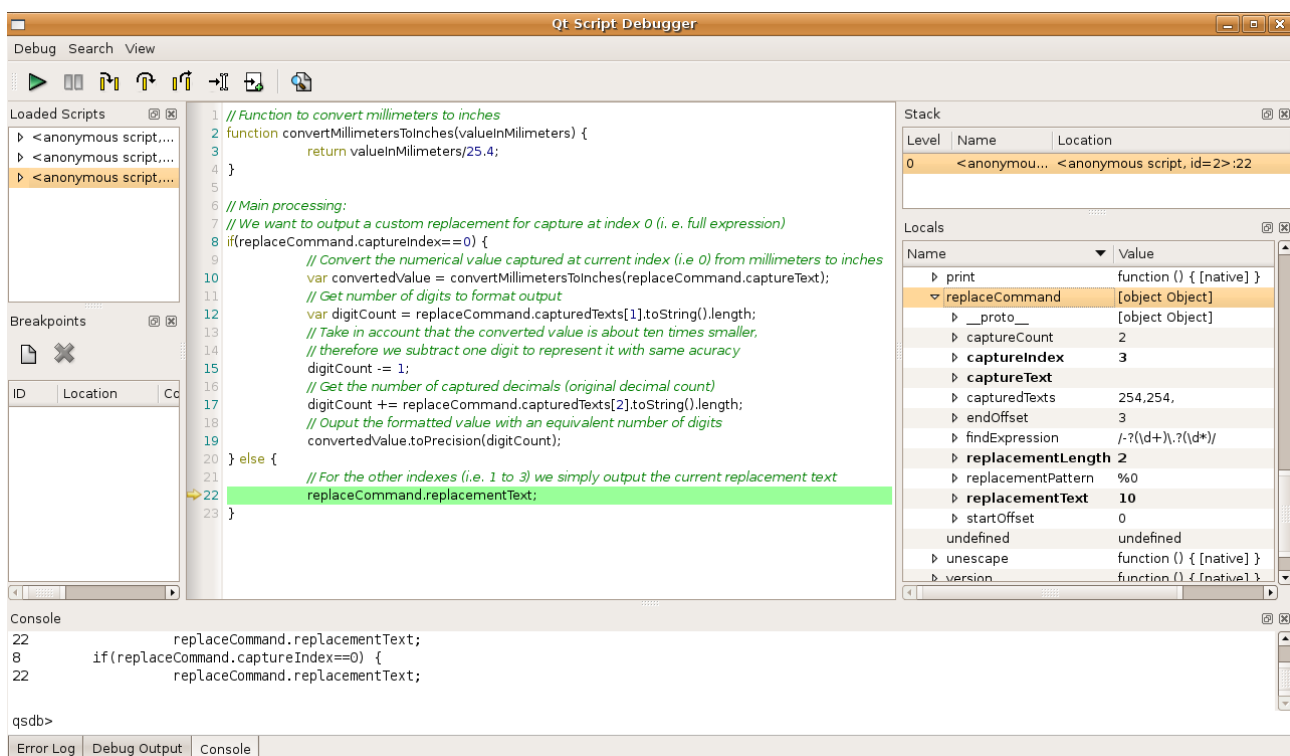
Script debugger in action. On the right we can see the 'Locals' dock window which displays the current context provided by the 'replaceCommand' object.

A user manual for script debugger is available at <u>http://doc.trolltech.com/4.7/qtscriptdebugger-manual.html</u>

8   Regular Expressions[1]

A regular expression, or "regexp", is a pattern for matching substrings in a text. This is useful in many contexts, e.g.:

Searching	A regexp provides more powerful pattern matching than simple substring matching, e.g., match one of the words *mail*, *letter* or *correspondence*, but none of the words *email*, *mailman*, *mailer*, *letterbox*, etc.
Search and Replace	A regexp can replace all occurrences of a substring with a different substring, e.g., replace all occurrences of &with & except where the *&* is already followed by an *amp;*.

The *Find & Replace It!* regexp is modeled on Perl's regexp language. It fully supports Unicode. The regexp can also be used in a simpler, **Wildcard** mode that is similar to the functionality found in command shells. The syntax rules used by regexp can be changed through the Syntax combo box. In particular, the pattern syntax can be set to **Simple text**, which means the pattern to be matched is interpreted as a plain string, i.e., special characters (e.g., backslash) are not escaped.

A good text on regexps is Mastering Regular Expressions (Third Edition) by Jeffrey E. F. Friedl, ISBN 0-596-52812-4.

8.1   Introduction

Regexps are built up from expressions, quantifiers, and assertions. The simplest expression is a character, e.g. x or 5. An expression can also be a set of characters enclosed in square brackets. **[ABCD]** will match an A or a B or a C or a D. We can write this same expression as **[A-D]**, and an experession to match any captital letter in the English alphabet is written as **[A-Z]**.

A quantifier specifies the number of occurrences of an expression that must be matched. **x{1,1}** means match one and only one x. **x{1,5}** means match a sequence of x characters that contains at least one x but no more than five.

Note that in general regexps cannot be used to check for balanced brackets or tags. For example, a regexp can be written to match an opening html ** and its closing **, if the ** tags are not nested, but if the ** tags are nested, that same regexp will match an opening ** tag with the wrong closing **. For the fragment *bold bolder*, the first ** would be matched with the first **, which is not correct. However, it is possible to write a regexp that will match nested brackets or tags correctly, but only if the number of nesting levels is fixed and known. If the number of nesting levels is not fixed and known, it is impossible to write a regexp that will not fail.

Suppose we want a regexp to match integers in the range 0 to 99. At least one digit is required, so we start with the expression **[0-9]{1,1}**, which matches a single digit exactly once. This regexp matches integers in the range 0 to 9. To match integers up to 99, increase the maximum number of occurrences to 2, so the regexp becomes **[0-9]{1,2}**. This regexp satisfies the original requirement to match integers from 0 to 99, but it will also match integers that occur in the middle of strings. If we want the matched integer to be the whole string, we must use the anchor assertions, **^** (caret) and **$** (dollar). When **^** is the first character in a regexp, it means the regexp must match from the beginning of the string. When **$** is the last character of the regexp, it means the regexp must match to the end of the string. The regexp becomes **^[0-9]{1,2}$**. Note that assertions, e.g. **^** and **$**, do not match characters but locations in the string.

If you have seen regexps described elsewhere, they may have looked different from the ones shown here. This is because some sets of characters and some quantifiers are so common that they have been given special symbols to represent them. **[0-9]** can be replaced with the symbol

---

1   This chapter is taken from the Qt ® documentation from Nokia ®, available under LGPL. It as been adapted to fit the purpose of this manual.

\d. The quantifier to match exactly one occurrence, **{1,1}**, can be replaced with the expression itself, i.e. **x{1,1}** is the same as **x**. So our 0 to 99 matcher could be written as **^\d{1,2}$**. It can also be written **^\d\d{0,1}$**, i.e. *From the start of the string, match a digit, followed immediately by 0 or 1 digits*. In practice, it would be written as **^\d\d?$**. The **?** is shorthand for the quantifier **{0,1}**, i.e. 0 or 1 occurrences. **?** makes an expression optional. The regexp **^\d\d?$** means *From the beginning of the string, match one digit, followed immediately by 0 or 1 more digit, followed immediately by end of string*.

To write a regexp that matches one of the words *mail* or *letter* or *correspondence* but does not match words that contain these words, e.g., *email, mailman, mailer,* and *letterbox,* start with a regexp that matches *mail.* Expressed fully, the regexp is **m{1,1}a{1,1}i{1,1}l{1,1}**, but because a character expression is automatically quantified by **{1,1}**, we can simplify the regexp to **mail**, i.e., an *m* followed by an *a* followed by an *i* followed by an *l*. Now we can use the vertical bar **|**, which means "*or*", to include the other two words, so our regexp for matching any of the three words becomes **mail|letter|correspondence**. Match *mail* or *letter* or *correspondence*. While this regexp will match one of the three words we want to match, it will also match words we don't want to match, e.g., *email.* To prevent the regexp from matching unwanted words, we must tell it to begin and end the match at word boundaries. First we enclose our regexp in parentheses, **(mail|letter|correspondence)**. Parentheses group expressions together, and they identify a part of the regexp that we wish to capture. Enclosing the expression in parentheses allows us to use it as a component in more complex regexps. It also allows us to examine which of the three words was actually matched. To force the match to begin and end on word boundaries, we enclose the regexp in **\b** *word boundary* assertions: **\b(mail|letter|correspondence)\b**. Now the regexp means: *Match a word boundary, followed by the regexp in parentheses, followed by a word boundary*. The **\b** assertion matches a *position* in the regexp, not a *character*. A word boundary is any non-word character, e.g., a space, newline, or the beginning or ending of a string.

If we want to replace ampersand characters with the HTML entity *&*, the regexp to match is simply *&*. But this regexp will also match ampersands that have already been converted to HTML entities. We want to replace only ampersands that are not already followed by *amp;*. For this, we need the negative lookahead assertion, **(?!___)**. The regexp can then be written as **&(?!amp;)**, i.e. *Match an ampersand that is **not** followed by* *amp;*.

If we want to count all the occurrences of *Eric* and *Eirik* in a string, two valid solutions are **\b(Eric|Eirik)\b** and **\bEi?ri[ck]\b**. The word boundary assertion **\b** is required to avoid matching words that contain either name, e.g. *Ericsson*. Note that the second regexp matches more spellings than we want: *Eric*, *Erik*, *Eiric* and *Eirik*.

Some of the examples discussed above are implemented in the examples section.

Regexps can match case insensitively using the **Case sensitive** check box, and can use non-greedy matching when the **Minimal match** mark is checked.

8.2   *Characters and Abbreviations for Sets of Characters*

Element	Meaning
c	A character represents itself unless it has a special regexp meaning. e.g. **c** matches the character *c*.
\c	A character that follows a backslash matches the character itself, except as specified below. e.g., To match a literal caret at the beginning of a string, write **\^**.
\a	Matches the ASCII bell (BEL, 0x07).
\f	Matches the ASCII form feed (FF, 0x0C).
\n	Matches the ASCII line feed (LF, 0x0A, Unix newline).
\r	Matches the ASCII carriage return (CR, 0x0D).
\t	Matches the ASCII horizontal tab (HT, 0x09).

Element	Meaning
\v	Matches the ASCII vertical tab (VT, 0x0B).
\x*hhhh*	Matches the Unicode character corresponding to the hexadecimal number *hhhh* (between 0x0000 and 0xFFFF).
\0*ooo* (i.e., \zero *ooo*)	matches the ASCII/Latin1 character for the octal number *ooo* (between 0 and 0377).
. (dot)	Matches any character (including newline).
\d	Matches a digit.
\D	Matches a non-digit.
\s	Matches a whitespace character including line separators.
\S	Matches a non-whitespace character.
\w	Matches a word character (letters, numbers, marks and '_').
\W	Matches a non-word character.
\n	The *n*-th backreference, e.g. \1, \2, etc.

8.3 Sets of Characters

Square brackets mean match any character contained in the square brackets. The character set abbreviations described above can appear in a character set in square brackets. Except for the character set abbreviations and the following two exceptions, characters do not have special meanings in square brackets.

^	The caret negates the character set if it occurs as the first character (i.e. immediately after the opening square bracket).*[abc]* matches *a* or *b* or *c*, but *[^abc]* matches anything but `a` or `b` or `c`.
-	The dash indicates a range of characters. *[W-Z]* matches `W` or `X` or `Y` or `Z`.

Using the predefined character set abbreviations is more portable than using character ranges across platforms and languages. For example, *[0-9]* matches a digit in Western alphabets but `\d` matches a digit in *any* alphabet.

Note: In other regexp documentation, sets of characters are often called "character classes".

8.4 Quantifiers

By default, an expression is automatically quantified by **{1,1}**, i.e. it should occur exactly once. In the following list, **E** stands for expression. An expression is a character, or an abbreviation for a set of characters, or a set of characters in square brackets, or an expression in parentheses.

E?	Matches zero or one occurrences of **E**. This quantifier means *The previous expression is optional*, because it will match whether or not the expression is found. **E?** is the same as **E{0,1}**. e.g., *dents?* matches `dent` or `dents`.
E+	Matches one or more occurrences of **E**. **E+** is the same as **E{1,}**. e.g., *0+* matches `0`, `00`, `000`, etc.
E*	Matches zero or more occurrences of *E*. It is the same as **E{0,}**. The * quantifier is often used in error where **+**should be used. For example, if `\s*$` is used in an expression to match strings that end in whitespace, it will match every string because `\s*$` means *Match zero or more whitespaces followed by end of string*. The correct regexp to match strings that have at least one trailing whitespace character is `\s+$`.
E{n}	Matches exactly *n* occurrences of **E**. **E{n}** is the same as repeating *E n* times. For example, *x{5}* is the same as *xxxxx*. It is also the same as **E{n,n}**, e.g. *x{5,5}*.
E{n,}	Matches at least *n* occurrences of **E**.
E{,m}	Matches at most *m* occurrences of **E**. **E{,m}** is the same as **E{0,m}.**

E{n,m}	Matches at least *n* and at most *m* occurrences of **E**.

To apply a quantifier to more than just the preceding character, use parentheses to group characters together in an expression. For example, `tag+` matches a `t` followed by an `a` followed by at least one `g`, whereas `(tag)+` matches at least one occurrence of `tag`.

Note: Quantifiers are normally "greedy". They always match as much text as they can. For example, `0+` matches the first zero it finds and all the consecutive zeros after the first zero. Applied to `20005`, it matches `20005`. Quantifiers can be made non-greedy through the check box **Minimal match**.

8.5 Capturing Text

Parentheses allow us to group elements together so that we can quantify and capture them. For example if we have the expression `mail|letter|correspondence` that matches a string we know that *one* of the words matched but not which one. Using parentheses allows us to "capture" whatever is matched within their bounds, so if we used `(mail|letter|correspondence)` and matched this regexp against the string `I sent you some email` we can use the `%x` replacement pattern to extract the matched characters, in this case `mail`.

We can use captured text within the regexp itself. To refer to the captured text we use "backreferences" which are indexed from 1, the same as for **%x**. For example we could search for duplicate words in a string using `\b(\w+)\W+\1\b` which means match a word boundary followed by one or more word characters followed by one or more non-word characters followed by the same text as the first parenthesized expression followed by a word boundary.

If we want to use parentheses purely for grouping and not for capturing we can use the non-capturing syntax, e.g. `(?:green|blue)`. Non-capturing parentheses begin `(?:` and end `)`. In this example we match either `green` or `blue` but we do not capture the match so we only know whether or not we matched but not which color we actually found. Using non-capturing parentheses is more efficient than using capturing parentheses since the regexp engine has to do less book-keeping.

Captured text can be accessed in replacement pattern using `%0` which returns the full matched expression, or using `%i` (with 1<i<=9) which returns the captured string at the given index.

Both capturing and non-capturing parentheses may be nested.

8.6 Assertions

Assertions make some statement about the text at the point where they occur in the regexp but they do not match any characters. In the following list **E** stands for any expression.

^	The caret signifies the beginning of the string. If you wish to match a literal `^` you must escape it by writing `\\^`. For example, `^#include` will only match strings which *begin* with the characters `#include`. (When the caret is the first character of a character set it has a special meaning, see Sets of Characters.)
$	The dollar signifies the end of the string. For example `\d\s*$` will match strings which end with a digit optionally followed by whitespace. If you wish to match a literal `$` you must escape it by writing `\\$`.
\b	A word boundary. For example the regexp `\bOK\b` means match immediately after a word boundary (e.g. start of string or whitespace) the letter `O` then the letter `K` immediately before another word boundary (e.g. end of string or whitespace). But note that the assertion does not actually match any whitespace so if we write `(\bOK\b)` and we have a match it will only contain `OK` even if the string is `It's OK now`.
\B	A non-word boundary. This assertion is true wherever `\b` is false. For example if we searched for `\Bon\B` in "Left on" the match would fail (space and end of string aren't

	non-word boundaries), but it would match in *tonne*.
(?=E)	Positive lookahead. This assertion is true if the expression matches at this point in the regexp. For example, **const(?=\s+char)** matches *const* whenever it is followed by *char*, as in *static const char **. (Compare with **const\s+char**, which matches *static const char **).
(?!E)	Negative lookahead. This assertion is true if the expression does not match at this point in the regexp. For example, **const(?!\s+char)** matches *const except* when it is followed by *char*.

8.7 Wildcard Matching

Most command shells such as *bash* or *cmd.exe* support "file globbing", the ability to identify a group of files by using wildcards. The Syntax combo box is used to switch between regexp and wildcard mode. Wildcard matching is much simpler than full regexps and has only four features:

c	Any character represents itself apart from those mentioned below. Thus **c** matches the character *c*.
?	Matches any single character. It is the same as **.** in full regexps.
*	Matches zero or more of any characters. It is the same as **.*** in full regexps.
[...]	Sets of characters can be represented in square brackets, similar to full regexps. Within the character class, like outside, backslash has no special meaning.

In the mode Wildcard, the wildcard characters cannot be escaped. In the mode Wildcard Unix, the character " escapes the wildcard.

For example if we are in wildcard mode and have strings which contain filenames we could identify HTML files with *.html. This will match zero or more characters followed by a dot followed by *h*, *t*, *m* and *l*.

Wildcard matching can be convenient because of its simplicity, but any wildcard regexp can be defined using full regexps, e.g. **.*\.html?$**. Notice that we can't match both *.html* and *.htm* files with a wildcard unless we use ***.htm*** which will also match *test.html.bak*. A full regexp gives us the precision we need, **.*\.html?$**.

8.8 Notes for Perl Users

Most of the character class abbreviations supported by Perl are supported by regexp's, see characters and abbreviations for sets of characters.

In regexps, apart from within character classes, **^** always signifies the start of the string, so carets must always be escaped unless used for that purpose. In Perl the meaning of caret varies automagically depending on where it occurs so escaping it is rarely necessary. The same applies to **$** which in regexps always signifies the end of the string.

Regexp's quantifiers are the same as Perl's greedy quantifiers. Non-greedy matching cannot be applied to individual quantifiers, but can be applied to all the quantifiers in the pattern. For example, to match the Perl regexp **ro+?m** requires: **ro+m** and ***Minimal match**=true*

The equivalent of Perl's **/i** option is ***Case sensitive*** check box turned on.

In regexp **.** matches any character, therefore all regexps have the equivalent of Perl's **/s** option. Regexp does not have an equivalent to Perl's **/m** option, but this can be emulated in various ways for example by splitting the input into lines or by looping with a regexp that searches for newlines.

Because regexp is string oriented, there are no **\A**, **\Z,** or **\z** assertions. The **\G** assertion is not supported but can be emulated in a loop.

Perl's **$&** is **%0**. There are no regexp equivalents for **$`**, **$'** or **$+**. Perl's capturing variables, *$1*, *$2*, ... correspond to **\1**, **\2** inside search pattern and **%1**, **%2** inside replacement pattern, etc.

Perl's extended **/x** syntax is not supported, nor are directives, e.g. **(?i)**, or regexp comments, e.g. **(?#comment)**.

Both zero-width positive and zero-width negative look-ahead assertions **(?=pattern)** and **(?! pattern)** are supported with the same syntax as Perl. Perl's look-behind assertions, "independent" sub-expressions and conditional expressions are not supported.

Non-capturing parentheses are also supported, with the same **(?:pattern)** syntax.

8.9 Examples

^\d\d?$
Match integers from 0 to 99

123	Do not match
-6	Do not match
6	Match

The third string matches *6*. This is a simple validation regexp for integers in the range 0 to 99.

^\S+$
Match strings without whitespaces

Hello world	Do not match
This_is-OK	Match

The second string matches *This_is-OK*. We've used the character set abbreviation **\S** (non-whitespace) and the anchors to match strings which contain no whitespace.

In the following example we match strings containing *mail* or *letter* or *correspondence* but only match whole words i.e. not *email*.

\b(mail|letter|correspondence)\b
Match words mail, letter *and* correspondence

I sent you an email	Do not match
Please write the letter	Match

The second string matches *Please write the letter*. The word *letter* is also captured (because of the parentheses). We can see what text we've captured like this: **%1 = \1 =** *letter*

This will capture the text from the first set of capturing parentheses (counting capturing left parentheses from left to right). The parentheses are counted from 1 since **%0** (**\0**) is the whole matched regexp (equivalent to **&** in most regexp engines).

&(?!amp;)
Match ampersands *but not* &

This & that	Match one occurrence at index 6
His & hers & theirs	Match one occurrence at index 16

9   Licensing Information

9.1 End User Licenses

The terms "This Software" below refer to *Find & Replace It!*.

9.1.1 End User License for NON App Store Customers

By installing and/or using this Software, you are confirming your acceptance of the Software and agreeing to become bound by the terms of the agreement as specified in the EULA: http://www.dprog.ch/multimedia/findreplaceit-docs/1.4/license_en.html.

Find & Replace It! is protected by an activation system. Hence to get a license for the full version of the product you should activate your product with a serial code called **Activation Key**. Without the activation, the software can still be run in demo mode with some limitations. You'll find more information in the Activation chapter.

The license will last forever, provided you do not reinstall your copy of the software on a different hardware configuration (see *Important Notes* below).

The license covers all minor updates (e.g.: all 0.x.y releases). Some major updates (e.g.: from 0.x to 1.x) might be included in the license but without guarantee.

Important Notes:

- One license allows you to install the software on only one computer at one time.
- A license is valid for only one type of operating system (i.e.: Linux, Mac or Windows).
- We kindly request people who have two computers running the same type of operating system (either Linux, Mac or Windows) to contact our support to get a second key for free.
- If your hardware configuration has changed, you might require an update of your license in order to reactivate the product. Such an update is totally free of charge.
- All demands regarding the licenses updates should be addressed to: support[at]dprog[dot]ch.

9.1.2 End User License for App Store Customers

If you buy this Software on the App Store from Apple, you must confirm your acceptance of the Software and agree to become bound by the terms of the agreement as specified in the EULA provided by Apple.

For people who bought *Find & Replace It!* on the App Store, there is no activation key required to use the software, but there is no demo version available either.

9.2   Third Party Licenses and Credits

This section contains information on the different licenses *Find & Replace It!* is shipped with, and credits to some of our many contributers.

This Software uses the Qt open source library from Nokia Corporation. This library is distributed under the GNU Lesser General Public License version 2.1 plus some exceptions (see http://doc.trolltech.com/4.7/lgpl.html). The Qt binaries and source code are available for download at http://qt.nokia.com/downloads. You are free to use your own versions of the Qt dynamic libraries instead of the ones shipped with the application. See the 'Customizing Install' section of http://www.dprog.ch/multimedia/findreplaceit-docs/1.4/install.html#title-customizing-install for more details. You are permitted to modify this Software for your own use only. Reverse engineering for debugging such modifications is allowed. For more details about licenses used by Qt please read the following links:

- http://doc.trolltech.com/4.7/licenses.html
- http://doc.trolltech.com/4.7/3rdparty.html

This Software uses the ICU open source library from IBM Corporation. ICU is a mature, widely used set of C/C++ and Java libraries providing Unicode and Globalization support for software applications. ICU is widely portable and gives applications the same results on all platforms and between C/C++ and Java software. ICU is released under a nonrestrictive open source license that is suitable for use with both commercial software and with other open source or free software (see http://source.icu-project.org/repos/icu/icu/trunk/license.html).

This Software uses the Syntax Highlighters plugins from Philippe Docourt. This is a collection of Qt plugins for managing syntax highlighting. It allows you to dynamically add syntax higlighting capabilities to your Qt applications. Currently it supports syntax highlighting for qscript, javascript, C++, Java, C#, and html/xml documents. This collection of plugins is distributed under the GNU Lesser General Public License version 2.1 (see http://www.gnu.org/licenses/lgpl-2.1.txt). The source code of thess plugins are available for download at http://www.dprog.net/joomla/products-topmenu-29/syntax-highlighters.

The Linux installer uses an auto-extractable archive generated by makeself. makeself.sh is a small shell script developped by Stéphane Peter that generates a self-extractable archive (see http://megastep.org/makeself).

The Windows installer is generated by NSIS (Nullsoft Scriptable Install System), an open source system to create Windows installers (see http://nsis.sourceforge.net/Main_Page).

This Software is built with the Qt Integration Scripts package from Philippe Docourt (see http://www.dprog.net/joomla/products-topmenu-29/qt-integration-scripts).

Most of the icons used in this Software are free icons, provided under either the GNU Lesser General Public License v2.1 (see http://www.gnu.org/licenses/lgpl-2.1.txt)or in the public domain. A special thanks to the authors:

- The Tango project members, authors of Tango icon theme: http://tango.freedesktop.org/Tango_Desktop_Project
- Everaldo Coelho, author of Christal icon collection: http://www.everaldo.com/crystal/
- David Vignoni, author of Nuvola icon set: http://www.icon-king.com/projects/nuvola/
- Nokia Corporation, author of Qt icons: http://qt.nokia.com/

9.3 Trademarks

- dProg, the dProg logo, *Find & Replace it!* and the *Find & Replace It!* logo are trademarks of dProg - Philippe Docourt in Switzerland.
- Nokia, the Nokia logo, Qt, and the Qt logo are trademarks of Nokia Corporation and/or its subsidiaries in Finland and other countries.
- Intel, Intel Inside (logos), MMX and Pentium are ® trademarks of Intel Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Linux is a ® trademark of Linus Torvalds in the United States, other countries or both.
- Mac, Mac OS, App Store and Macintosh are ® trademarks of Apple Computer, Inc., registered in the U.S. and other countries.
- Microsoft, Windows, Windows NT, XP, Visual Studio, Word and the Windows logo are ® trademarks of Microsoft Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- All other company, product, or service names may be trademarks or service marks of others and are the property of their respective owners.